UNIT I

1 Understanding Big Data

Syllabus

Introduction to big data - convergence of key trends - unstructured data - industry examples of big data - web analytics - big data applications- big data technologies - introduction to Hadoop - open source technologies - cloud and big data - mobile business intelligence - Crowd sourcing analytics - inter and trans firewall analytics.

Contents

1.1 Introduction to Big Data

- Big data can be defined as very large volumes of data available at various sources, in varying degrees of complexity, generated at different speed i.e., velocities and varying degrees of ambiguity, which cannot be processed using traditional technologies, processing methods, algorithms or any commercial off-the-shelf solutions.
- 'Big data' is a term used to describe a collection of data that is huge in size and yet growing exponentially with time. In short, such data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently.
- The processing of big data begins with the raw data that isn't aggregated or organized and is most often impossible to store in the memory of a single computer.
- Big data processing is a set of techniques or programming models to access large-scale data to extract useful information for supporting and providing decisions. Hadoop is the open-source implementation of MapReduce and is widely used for big data processing.

1.1.1 Difference between Data Science and Big Data

Sr. No.	Data science	Big data
1.	It is a field of scientific analysis of data in order to solve analytically complex problems and the significant and necessary activity of cleansing, preparing of data.	Big data is storing and processing large volume of structured and unstructured data that can not be possible with traditional applications.
2.	It is used in Biotech, energy, gaming and insurance.	Used in retail, education, healthcare and social media.
3.	Goals : Data classification, anomaly detection, prediction, scoring and ranking.	Goals : To provide better customer service, identifying new revenue opportunities, effective marketing etc.

1.1.2 Benefits of Big Data Processing

Benefits of big data processing :

1. Improved customer service.
2. Business can utilize outside intelligence while taking decisions.
3. Reducing maintenance costs.

4. Re-develop your products : Big data can also help you understand how others perceive your products so that you can adapt them or your marketing, if need be.
5. Early identification of risk to the product / services, if any.
6. Better operational efficiency.

1.1.3 Big Data Challenges

- Collecting, storing and processing big data comes with its own set of challenges :
  1. Big data is growing exponentially and existing data management solutions have to be constantly updated to cope with the three Vs.
  2. Organizations do not have enough skilled data professionals who can understand and work with big data and big data tools.

1.2 Convergence of Key Trends

- The essence of computer applications is to store things in the real world into computer systems in the form of data, i.e., it is a process of producing data. Some data are the records related to culture and society and others are the descriptions of phenomena of the universe and life. The large scale of data is rapidly generated and stored in computer systems, which is called data explosion.
- Data is generated automatically by mobile devices and computers, think facebook, search queries, directions and GPS locations and image capture.
- Sensors also generate volumes of data, including medical data and commerce location-based sensors. Experts expect 55 billion IP - enabled sensors by 2021. Even storage of all this data is expensive. Analysis gets more important and more expensive every year.
- Fig. 1.2.1 shows the big data explosion by the current data boom and how critical it is for us to be able to extract meaning from all of this data.

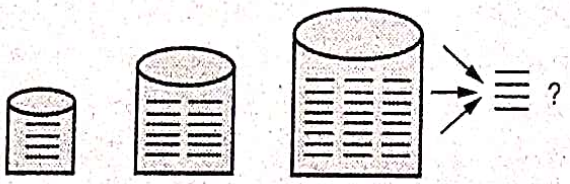


Fig. 1.2.1 Data explosion

- The phenomena of exponential multiplication of data that gets stored is termed as "Data Explosion". Continuous inflow of real-time data from various processes, machinery and manual inputs keeps flooding the storage servers every second.
- Sending emails, making phone calls, collecting information for campaigns; each day we create a massive amount of data just by going about our normal business

and this data explosion does not seem to be slowing down. In fact, 90 % of the data that currently exists was created in just the last two years.

- Reason for this data explosion is **Innovation.**
  1. **Business model transformation :** Innovation changed the way in which we do business, provide services. The data world is governed by three fundamental trends are business model transformation, globalization and personalization of services.
     - Organizations have traditionally treated data as a legal or compliance requirement, supporting limited management reporting requirements. Consequently, organizations have treated data as a cost to be minimized.
     - The businesses are required to produce more data related to product and provide services to cater each sector and channel of customer.
  2. **Globalization :** Globalization is an emerging trend in business where organizations start operating on an international scale. From manufacturing to customer service, globalization has changed the commerce of the world. Variety and different formats of data are generated due to globalization.
  3. **Personalization of services :** To enhance customer service, the form of one-to-one marketing in the form of personalization of service is opted by the customer. Customers expect communication through various channels increases the speed of data generation.
  4. **New sources of data :** The shift to online advertising supported by the likes of Google, Yahoo and others is a key driver in the data boom. Social media, mobile devices, sensor networks and new media are on the fingertips of customers or users. The data generated through this is used by corporations for decision support systems like business intelligence and analytics. The growth of technology helped to emerge new business models over the last decade or more. Integration of all the data across the enterprise is used to create business decision support platform.

1.2.1 V's of Big Data

- We differentiate big data characteristics from traditional data by one or more of the five V's : *Volume, velocity, variety, veracity and value.*
  1. **Volume :** Volumes of data are larger than that conventional relational database infrastructure can cope with. It consisting of terabytes or petabytes of data.
     - Fig. 1.2.2 shows big data volume.

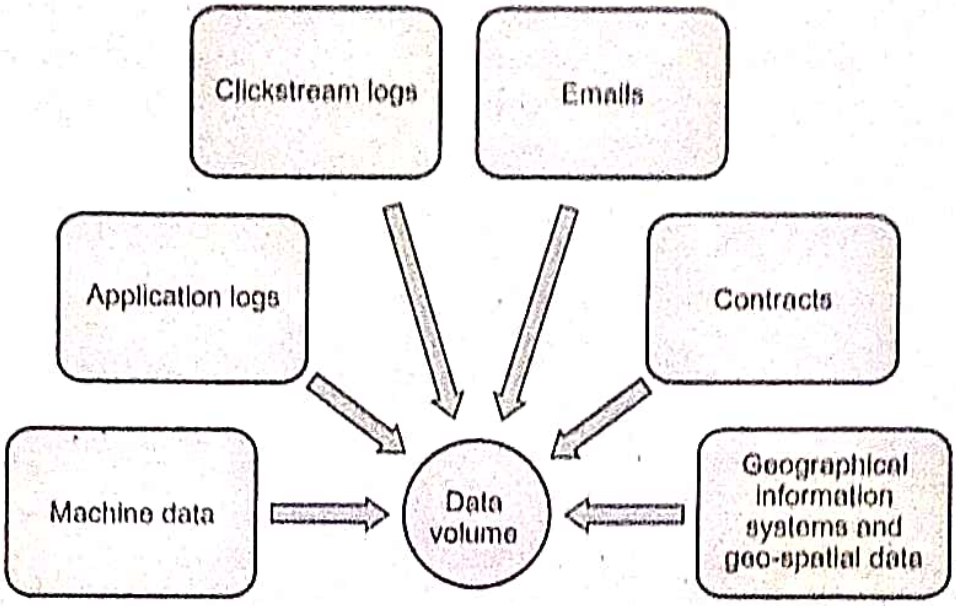


Fig. 1.2.2 Big data volumo

2. **Velocity :** The term 'velocity' refers to the speed of generation of data. How fast the data is generated and processed to meet the demands, determines real potential in the data. It is being created in or near real-time.
3. **Variety :** It refers to heterogeneous sources and the nature of data, both structured and unstructured.
   - Fig. 1.2.3 (a) and Fig. 1.2.3 (b) shows big data velocity and data variety.

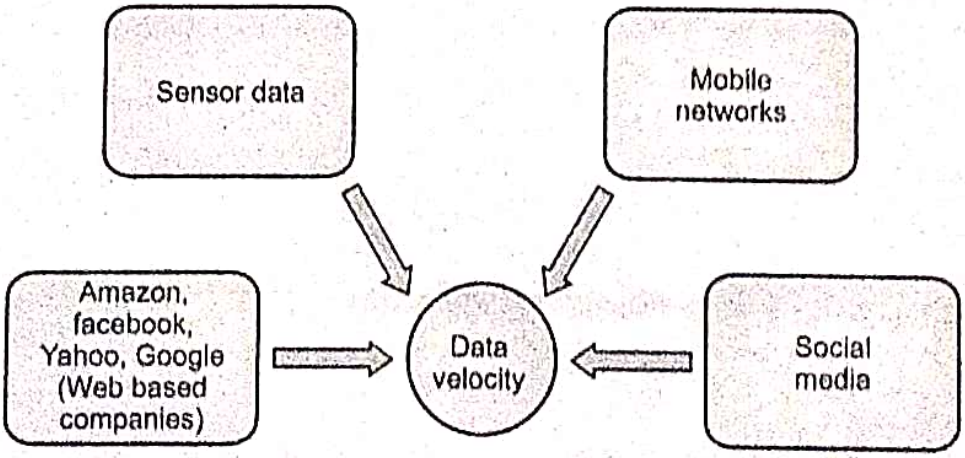


Fig. 1.2.3 (a) Data velocity

(Refer Fig. 1.2.3 (b) on next page)

4. **Value :** It represents the business value to be derived from big data.
   - The ultimate objective of any big data project should be to generate some sort of value for the company doing all the analysis. Otherwise, you're just performing some technological task for technology's sake.

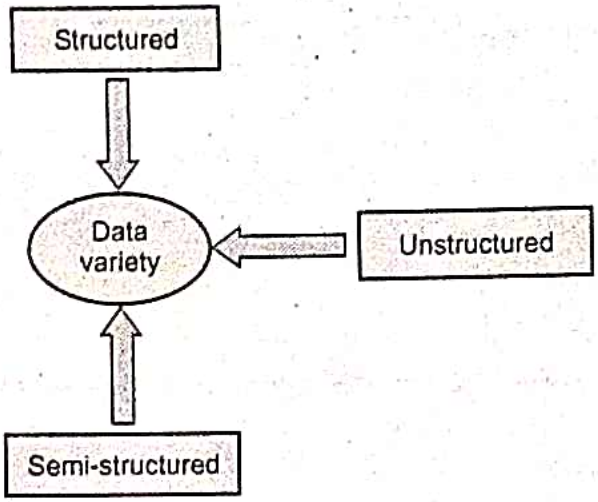


Fig. 1.2.3 (b) Data variety

- For real-time spatial big data, decisions can be enhanced through visualization of dynamic change in such spatial phenomena as climate, traffic, social-media-based attitudes and massive inventory locations.
- Exploration of data trends can include spatial proximities and relationships. Once spatial big data are structured, formal spatial analytics can be applied, such as spatial autocorrelation, overlays, buffering, spatial cluster techniques and location quotients.

5. **Veracity :** Big data must be fed with relevant and true data. We will not be able to perform useful analytics if much of the incoming data comes from false sources or has errors. Veracity refers to the level of trustiness or messiness of data and if higher the trustiness of the data, then lower the messiness and vice versa. It relates to the assurance of the data's quality, integrity, credibility and accuracy. We must evaluate the data for accuracy before using it for business insights because it is obtained from multiple sources.

1.2.2 Compare Cloud Computing and Big Data

Sr. No.	Cloud computing	Big data
1.	It provides resources on demand.	It provides a way to handle huge volumes of data and generate insights.
2.	It refers to internet services from SaaS, PaaS to IaaS.	It refers to data, which can be structured, semi-structured or unstructured.
3.	Cloud is used to store data and information on remote servers.	It is used to describe a huge volume of data and information.
4.	Cloud computing is economical as it has low maintenance costs centralized platform no upfront cost and disaster safe implementation.	Big data is a highly scalable, robust ecosystem and cost-effective.

5.	Vendors and solution providers of cloud computing are Google, Amazon web service, Dell, Microsoft, Apple and IBM.	Vendors and solution providers of big data are Cloudera, Hortonworks, Apache and MapR.
6.	The main focus of cloud computing is to provide computer resources and services with the help of network connection.	Main focus of big data is about solving problems when a huge amount of data generating and processing.

1.3 Unstructured Data

- Unstructured data is data that does not follow a specified format. Row and columns are not used for unstructured data. Therefore it is difficult to retrieve required information. Unstructured data has no identifiable structure.
- For example of unstructured data is e-mails, click streams, textual data, images, log data and videos.
- In the case of unstructured data, the size is not the only problem, deriving value or getting results out of unstructured data is much complex and challenging as compared of structured data.
- The unstructured data can be in the form of text : (Documents, email messages, customer feedbacks), audio, video, images. Email is an example of unstructured data.
- Even today in most of the organizations more than 80 % of the data are in unstructured form. This carries lots of information. But extracting information from these various sources is a very big challenge.
- Characteristics of unstructured data :
  1. There is no a structural restriction or binding for the data.
  2. Data can be of any type.
  3. Unstructured data does not follow any structural rules.
  4. There are no predefined formats, restriction or sequence for unstructured data.
  5. Since there is no structural binding for unstructured data it is unpredictable in nature.

Examples of machine generated unstructured data :

1. **Satellite images :** This includes weather data or the data that the government captures in its satellite surveillance imagery.
2. **Scientific data :** This includes atmospheric data and high energy physics.
3. **Photographs and video :** This include security, surveillance and traffic video.

Structured Data :

- Structured data is arranged in rows and column format. It helps for application to retrieve and process data easily. Database management system is used for storing structured data.
- Any data that can be stroed in the form of a particular fixed is known as structured data. For example, data stored in the colums and rows of tables in relational database management systems is a form of structured data.

1.3.1 Difference between Structured and Unstructured Data

Parameters	Structured data	Unstructured data
Representation	It is in discrete form. i.e. stored in row and column format.	Unstructured data is data that does not follow a specified format.
Metadata	Syntax	Semantics
Storage	Database management system	Unmanaged file structure
Standard	SQL, ADO.net, ODBC	Open XML, SMTO, SMS
Tools for integration	ETL	Batch processing or manual data entry.
characteristics	With a structured document, certain information always appears in the same location on the page.	In unstructured document information can appear in unexpected places on the document.
Used by organizations	Low volume operations	High volume operations

1.4 Industry Examples of Big Data

- Big data plays an important role in digital marketing. Each day information shared digitally increases significantly. With the help of big data, marketers can analyze every action of the consumer. It provides better marketing insights and it helps marketers to make more accurate and advanced marketing strategies.
- Reasons why big data is important for digital marketers :
  a) Real-time customer insights
  b) Personalized targeting
  c) Increasing sales
  d) Improves the efficiency of a marketing campaign
  e) Budget optimization
  f) Measuring campaign's results more accurately.

- Data constantly informs marketing teams of customer behaviors and industry trends and is used to optimize future efforts, create innovative campaigns and build lasting relationships with customers.
- Big data regarding customers provides marketers details about user demographics, locations and interests, which can be used to personalize the product experience and increase customer loyalty over time.
- Big data solutions can help organize data and pinpoint which marketing campaigns, strategies or social channels are getting the most traction. This lets marketers allocate marketing resources and reduce costs for projects that are not yielding as much revenue or meeting desired audience goals.
- Personalized targeting : Nowadays, personalization is the key strategy for every marketer. Engaging the customers at the right moment with the right message is the biggest issue for marketers. Big data helps marketers to create targeted and personalized campaigns.
- Personalized marketing is creating and delivering messages to the individuals or the group of the audience through data analysis with the help of consumer's data such as geolocation, browsing history, clickstream behavior and purchasing history. It is also known as **one - to - one marketing**.
- Consumer insights : In this day an age, marketing has become the ability of a company to interpret the data and change its strategies accordingly. Big data allows for real-time consumer insights which is crucial to understanding the habits of your customers. By interacting with your consumers through social media you will know exactly what they want and expect from your product or service, which will be key to distinguishing your campaign from your competitors.
- Help increase sales : Big data will help with demand predictions for a product or service. Information gathered on user behaviour will allow marketers to answer what types of product their users are buying, how often they conduct purchases or search for a product or service and lastly, what payment methods they prefer using.
- Analyse campaign results : Big data allows marketers to measure their campaign performance. This is the most important part of digital marketing. Marketers will use reports to measure any negative changes to marketing KPIs. If they have not achieved the desired results it will be a signal that the strategy would need to be changed in order to maximize revenue and make your marketing efforts more scalable in future.

1.5 Web Analytics

- Web analytics is the collection, reporting and analysis of website data. The focus is on identifying measures based on your organizational and user goals and using the website data to determine the success or failure of those goals and to drive strategy and improve the user's experience.
- The WWW is an evolving system for publishing and accessing resources and services across the Internet. The web is an open system. Its operations are based on freely published communication standards and documents standards.
- Web analytics is important to help us to :
  1. Refine your marketing campaigns
  2. Understand your website visitors
  3. Analyze website conversions
  4. Improve the website user experience
  5. Boost your search engine ranking
  6. Understand and optimize referral sources
  7. Boost online sales.
- Businesses use web analytics platforms to measure and benchmark site performance and to look at key performance indicators that drive their business, such as purchase conversion rate.
- Website analytics provide insights and data that can be used to create a better user experience for website visitors. Understanding customer behavior is also key to optimizing a website for key conversion metrics.
- For example, web analytics will show us the most popular pages on your website, and the most popular paths to purchase. With website analytics, we can also accurately track the effectiveness of your online marketing campaigns to help inform future efforts.
- Web analytics can help a digital marketer understand their customers better by providing :
  1. Insight into who the customers are and their interests
  2. Conversion challenges
  3. Enhanced appreciation of what consumers like or do not like
  4. Understanding of how to improve user experience for the consumer.

1.6 Big Data Applications

- Big data applications can help companies to make better business decisions by analyzing large volumes of data and discovering hidden patterns. These data sets might be from social media, data captured by sensors, website logs, customer feedbacks, etc. Organizations are spending huge amounts on big data applications to discover hidden patterns, unknown associations, market style, consumer preferences and other valuable business information.
- Domains where big data can be applied to health care, media and entertainment, IoT, manufacturing and government.
- **Relation between IIoT and Big Data :** Big data production in the industrial Internet of Things (IIoT) is evident due to the massive deployment of sensors and Internet of Things (IoT) devices. However, big data processing is challenging due to limited computational, networking and storage resources at IoT device-end. Big Data Analytics (BDA) is expected to provide operational and customer-level intelligence in IIoT systems.
- The extensive installation of sensors on machines causes a massive increase in the volume of data collected within industrial processes. The data consist of operating data, error lists, history of maintenance activities and alike.
- In combination with the related business data, the overall plethora of data provides the raw material for process optimizations and other applications. To set this potential for optimizations free, the raw data needs to be processed systematically, passing through various algorithms.
- The results are prepared information with specific application objectives. Especially pattern detection is to mention in this context, since this method identifies and quantifies cause and effect correlations and allows predictions of state changes. The significance of the information given out by the analysis depends on the amount of data processed.

1. Healthcare :

- Big data analytics for healthcare uses health-related information of an individual or community to understand a patient, organization or community. In the past, managing and analyzing healthcare data was tedious and expensive. More recently, technology has helped the healthcare sector make leaps and bounds to keep up with the flow of big data in healthcare.
- Diagnostic devices, medical machinery, instrumentation, online services sources such as these are transferring data throughout a healthcare network. This is done with the help of big data tools such as Hadoop and Spark.

- One of the most current and relevant big data examples in healthcare is how it has impacted the global coronavirus crisis. Big data analytics for healthcare supported the rapid development of COVID-19 vaccines. Researchers can share data with each other to develop advanced medications very quickly. Big data in healthcare also predicted the spread of disease by allowing healthcare information to be processed much more rapidly than in the past during other pandemics.
- Smoother hospital administration : Healthcare administration becomes much smoother with the help of big data. It helps to reduce the cost of care measurement, provide the best clinical support and manage the population of at-risk patients. It also helps medical experts analyze data from diverse sources. It helps healthcare providers conclude the deviations among patients and the effects treatments have on their health.
- Fraud prevention and detection : Big data helps to prevent a wide range of errors on the side of health administrators in the form of wrong dosage, wrong medicines and other human errors. It will also be particularly useful to insurance companies. They can prevent a wide range of fraudulent claims of insurance.
- **Challenges of big data in healthcare :** As a relatively new field, big data in healthcare is still evolving to keep up with the fast pace and changing nature of technology. With such vast amounts of data available to work with, organizations and leaders can struggle with knowing where and how to start with data analytics in healthcare to find the information that is meaningful.
- Many healthcare organizations lack adequate systems and databases and the skilled professionals to handle them. As such, the demand for healthcare analysts with advanced education and training is very high in the World.

2. Manufacturing :

- Improving efficiency across the business helps a manufacturing company control costs, increase productivity, and boost margins. Automated production lines are already standard practice for many, but manufacturing big data can exponentially improve line speed and quality.
- Manufacturing big data also increases transparency into the entire supply chain-for example, by using sensor and RFID data to track the location of tools, parts and inventory in real time, reducing interruptions and delays.
- Companies can also increase supply chain transparency by analyzing individual processes and their interdependencies for opportunities to optimize everything from demand forecasting and inventory management to price optimization.
- Speeding up assembly : Part of the key to manufacturing more products is to simply make the whole process quicker. With big data, manufacturers have been able to segment their production to identify which parts of the process go the

fastest. Knowing which products are faster and easier to produce can help companies know where to focus their efforts, perhaps even concentrating solely on those products for maximum production. It helps for companies to know where they are most efficient, with the added possibility of working on those areas that need the most improvement.

- AI-driven analysis of manufacturing big data enables companies to aggregate and analyze both their own and competitor's pricing and cost data to produce continually optimized price variants. For manufacturers that focus on build-to-order products, ML can also ensure the accuracy of their customized configurations and streamline the Configure-Price-Quote (CPQ) workflow.

1.7 Big Data Technologies

- Big data technology is defined as the technology and a software utility that is designed for analysis, processing and extraction of the information from a large set of extremely complex structures and large data sets which is very difficult for traditional systems to deal with. Big data technology is used to handle both real-time and batch related data.
- Big data technology is defined as **software-utility**. This technology is primarily designed to analyze, process and extract information from a large data set and a huge set of extremely complex structures. This is very difficult for traditional data processing software to deal with.
- Big data technologies including Apache Hadoop, Apache Spark, MongoDB, Cassandra, Plotly, Pig, Tableau and Apache Cassandra etc.
- Cassandra : Cassandra is one of the leading big data technologies among the list of top NoSQL databases. It is open-source, distributed and has extensive column storage options. It is freely available and provides high availability without fail.
- Apache Pig is a high - level scripting language used to execute queries for larger datasets that are used within Hadoop.
- Apache Spark is a fast, in - Memory data processing engine suitable for use in a wide range of circumstances. Spark can be deployed in several ways, it features java, Python, Scala and R programming languages and supports SQL, streaming data, machine learning and graph processing, which can be used together in an application.
- MongoDB : MongoDB is another important component of big data technologies in terms of storage. No relational properties and RDBMS properties apply to MongoDb because it is a NoSQL database. This is not the same as traditional RDBMS databases that use structured query languages. Instead, MongoDB uses schema documents.

1.8 Introduction to Hadoop

- Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Hadoop is designed to scale up from a single computer to thousands of clustered computers, with each machine offering local computation and storage.
- While Hadoop is sometimes referred to as an acronym for **High Availability Distributed Object Oriented Platform**.
- The Hadoop framework consists of a storage layer known as the **Hadoop Distributed File System (HDFS)** and a processing framework called the **MapReduce programming model.** Hadoop splits large amounts of data into chunks, distributes them within the network cluster and processes them in its MapReduce Framework.
- Hadoop can also be installed on cloud servers to better manage the compute and storage resources required for big data. Leading cloud vendors such as Amazon Web Services (AWS) and Microsoft Azure offer solutions. Cloudera supports Hadoop workloads both on-premises and in the cloud, including options for one or more public cloud environments from multiple vendors.
- Hadoop provides a distributed file system and a framework for the analysis and transformation of very large data sets using the MapReduce paradigm. An important characteristic of Hadoop is the partitioning of data and computation across many (thousands) of hosts and executing application computations in parallel close to their data. A Hadoop cluster scales computation capacity, storage capacity and I/O bandwidth by simply adding commodity servers.
- **Key features of Hadoop :**
  1. Cost Effective System
  2. Large Cluster of Nodes
  3. Parallel Processing
  4. Distributed Data
  5. Automatic Failover Management
  6. Data Locality Optimization
  7. Heterogeneous Cluster
  8. Scalability.
- Hadoop allows for the distribution of datasets across a cluster of commodity hardware. Processing is performed in parallel on multiple servers simultaneously. Software clients input data into Hadoop. HDFS handles metadata and the

distributed file system. MapReduce then processes and converts the data. Finally, YARN divides the jobs across the computing cluster.

- All Hadoop modules are designed with a fundamental assumption that hardware failures of individual machines or racks of machines are common and should be automatically handled in software by the framework.
- Challenges of Hadoop :

  MapReduce complexity : As a file-intensive system, MapReduce can be a difficult tool to utilize for complex jobs, such as interactive analytical tasks.
- There are four main libraries in Hadoop.
  1. **Hadoop Common :** This provides utilities used by all other modules in Hadoop.
  2. **Hadoop MapReduce :** This works as a parallel framework for scheduling and processing the data.
  3. **Hadoop YARN :** This is an acronym for **Yet Another Resource Navigator.** It is an improved version of MapReduce and is used for processes running over Hadoop.
  4. **Hadoop Distributed File System - HDFS :** This stores data and maintains records over various machines or clusters. It also allows the data to be stored in an accessible format.

1.8.1 Hadoop Ecosystem

- Hadoop ecosystem is neither a programming language nor a service, it is a platform or framework which solves big data problems.
- The Hadoop ecosystem refers to the various components of the Apache Hadoop software library, as well as to the accessories and tools provided by the Apache Software Foundation for these types of software projects and to the ways that they work together.
- Hadoop is a Java - based framework that is extremely popular for handling and analysing large sets of data. The idea of a Hadoop ecosystem involves the use of different parts of the core Hadoop set such as MapReduce, a framework for handling vast amounts of data and the Hadoop Distributed File System (HDFS), a sophisticated file - handling system. There is also YARN, a Hadoop resource manager.
- In addition to these core elements of Hadoop, Apache has also delivered other kinds of accessories or complementary tools for developers.
- Some of the most well - known tools of the Hadoop ecosystem include HDFS, Hive, Pig, YARN, MapReduce, Spark, HBase, Oozie, Sqoop, Zookeeper, etc.

- Fig. 1.8.1 shows Apache Hadoop ecosystem.

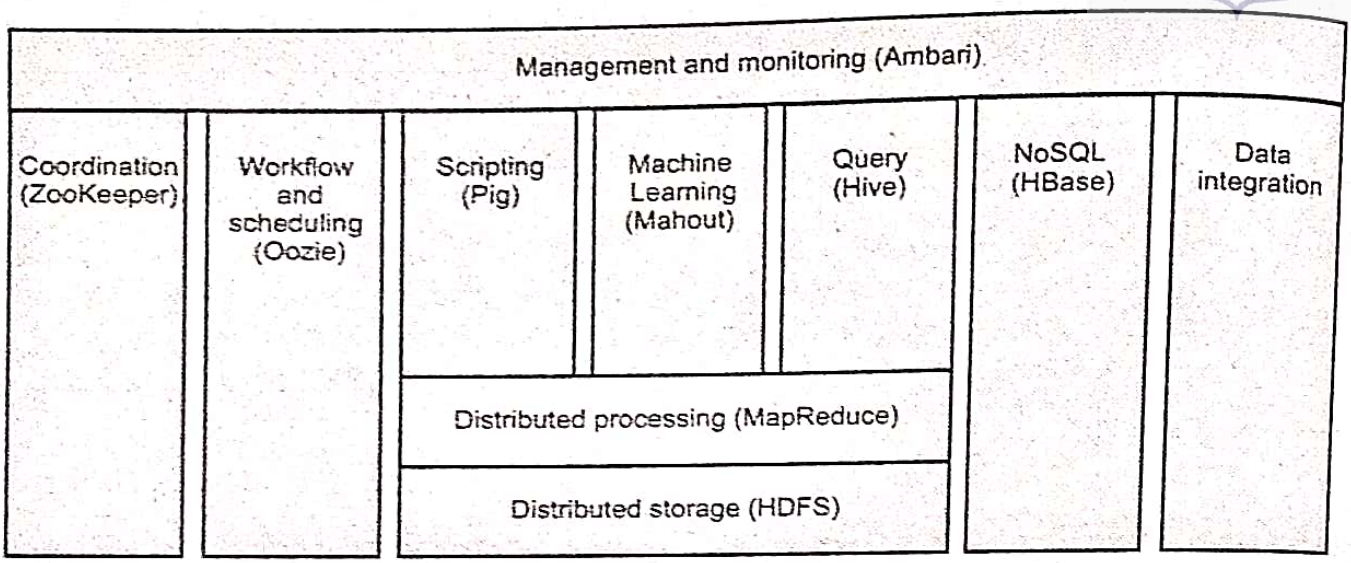


Fig. 1.8.1 Apache Hadoop ecosystem

- Hadoop Distributed File System (HDFS), is one of the largest Apache projects and primary storage system of Hadoop. It employs a NameNode and DataNode architecture. It is a distributed file system able to store large files running over the cluster of commodity hardware.
- YARN stands for Yet Another Resource Negotiator. It is one of the core components in open source Apache Hadoop suitable for resource management. It is responsible for managing workloads, monitoring and security controls implementation.
- Hive is an ETL and Data warehousing tool used to query or analyze large datasets stored within the Hadoop ecosystem. Hive has three main functions : Data summarization, query and analysis of unstructured and semi - structured data in Hadoop.
- Map - Reduce : It is the core component of processing in a Hadoop Ecosystem as it provides the logic of processing. In other words, MapReduce is a software framework which helps in writing applications that processes large data sets using distributed and parallel algorithms inside Hadoop environment.
- Apache Pig is a high - level scripting language used to execute queries for larger datasets that are used within Hadoop.
- Apache Spark is a fast, in - memory data processing engine suitable for use in a wide range of circumstances. Spark can be deployed in several ways, it features Java, Python, Scala and R programming languages and supports SQL, streaming

data, machine learning and graph processing, which can be used together in an application.

- Apache HBase is a Hadoop ecosystem component which is a distributed database that was designed to store structured data in tables that could have billions of rows and millions of columns. HBase is scalable, distributed and NoSQL database that is built on top of HDFS. HBase provide real - time access to read or write data in HDFS.

1.8.2 Hadoop Advantages

1. Scalable : Hadoop cluster can be extended by just adding nodes in the cluster.
2. Cost effective : Hadoop is open source and uses commodity hardware to store data so it is really cost effective as compared to traditional relational database management systems.
3. Resilient to failure : HDFS has the property with which it can replicate data over the network.
4. Hadoop can handle unstructured as well as semi-structured data.
5. The unique storage method of Hadoop is based on a distributed file system that effectively maps data wherever the cluster is located.

1.9 Open Source Technologies

- *Open source software* is like any other software (closed/proprietary software). This software is differentiated by its use and licenses. Open source software guarantees the right to access and modify the source code and to use, reuses and redistribute the software, all with no royalty or other costs.
- Standard Software is sold and supported commercially. However, Open Source software can be sold and/or supported commercially, too. Open source is a disruptive technology.
- *Open source* is an approach to the design, development and distribution of software, offering practical accessibility to software's source code.
- Open source licenses must permit non-exclusive commercial exploitation of the licensed work, must make available the work's source code and must permit the creation of derivative works from the work itself.
- The Netscape Public License and subsequently under the Mozilla Public License.
- **Proprietary software** is computer software which is the legal property of one party. The terms of use for other parties are defined by contracts or licensing agreements. These terms may include various privileges to share, alter, dissemble, and use the software and its code.

- **Closed source** is a term for software whose license does not allow for the release or distribution of the software's source code. Generally, it means only the binaries of a computer program are distributed and the license provides no access to the program's source code. The source code of such programs is usually regarded as a trade secret of the company. Access to source code by third parties commonly requires the party to sign a non-disclosure agreement.

Need of open source

- The demands of consumers as well as enterprises are ever increasing with the increase in the information technology usage. Information technology solutions are required to satisfy their different needs. It is a fact that a single solution provider cannot produce all the needed solutions. Open source, freeware and free software are now available for anyone and for any use.
- In the 1970s and early 1980s, the software organization started using technical measures to prevent computer users from being able to study and modify software. The copyright law was extended to computer programs in 1980. The free software movement was conceived in 1983 by Richard Stallman to satisfy the need for and to give the benefit of "software freedom" to computer users.
- Richard Stallman declared the idea of the GNU operating system in September 1983. The GNU Manifesto was written by Richard Stallman and published in March 1985.
- The Free Software Foundation (FSF) is a non-profit corporation started by Richard Stallman on 4 October 1985 to support the free software movement, a copyleft based movement which aims to promote the universal freedom to distribute and modify computer software without restriction. In February 1986, the first formal definition of free software was published.
- The term "free software" is associated with FSFs definition, and the term "open source software" is associated with OSI's definition. FSFs and OSI's definitions are worded quite differently but the set of software that they cover is almost identical.
- One of the primary goals of this foundation was the development of a free and open computer operating system and application software that can be used and shared among different users with complete freedom.
- While open source differs from the operation of traditional copyright licensing by permitting both open distribution and open modification.
- Before the term open source became widely adopted, developers and producers used a variety of phrases to describe the concept. The term open source gained popularity with the rise of the Internet, which provided access to diverse production models, communication paths and last but not least, interactive communities.

- Netscape licensed and released its code as open source under **Definition of Open Source Software.**

Successes of open source

- Successful open source projects make up many of today's most widely used technologies

  Operating systems : Linux, Symbian, GNU Project, NetBSD.

  Servers : Apache, Tomcat, MediaWiki, Drupal, WordPress, Eclipse, Moodle, Joomla

  Programming languages : Java, JavaScript, PHP, Python, Ruby.

  Client software : Mozilla Firefox, Mozilla Thunderbird, OpenOffice, Songbird, Audacity, 7-Zip.

  Digital content : Wikipedia, Wiktionary, Project Gutenberg.

Examples in open source and propritary software :

Classification of software	Open Source software	Propritary software
Operating systems	Linux	MS Windows, XP, Vista ; SUN Solaris
Word processing and office applications	openOffice	MS Office, Adobe Framemaker
Software development	Eclipse, JDK	MS Visual Studio, .net
Multimedia content creation	Gimp	Adobe Photoshop
Web page design	Typo 3	MS Frontpage, Adobe Flash, Dreamweaver

1.9.1 Difference between Open Source and Open Standards

- **Open source software** is a type of software where the user has access to the software's source code and can freely use, modify and distribute the software. Thus open source concerns the code the software is made of.
- **Open standards** denotes that the code responsible for communication with other systems is open and has technical specifications which are accessible free of charge. Thus open standards concern the communication between software.

1.9.2 Advantages of Open Sources

1. The right to use the software in any way.
2. There is usually no license cost and free of cost.
3. The source code is open and can be modified freely.

4. It is possible to reuse the software in another context or with another public authority.
5. Open standards.
6. It provides higher flexibility.

1.9.3 Disadvantages of Open Sources

1. There is no guarantee that development will happen.
2. It is sometimes difficult to know that a project exists, and its current status.
3. No secured follow-up development strategy.

1.9.4 Application of Open Source Software

- Following is the list of applications where open source software is used.
  1. Social networking
  2. Multimedia
  3. Animation
  4. Accounting
  5. Instant messaging
  6. ERP
  7. Desktop publishing
  8. Website development
  9. Resource management
  10. Video editing.

1.9.5 Comparison of Open Source with Close Source / Proprietary Software

Sr. No.	Open source software	Close source / proprietary software
1.	Source code freely available.	Source code is kept secret.
2.	Modification are allowed.	Modifications are not allowed.
3.	Licenses may do their own development.	All upgrades, support, maintenance and development are done by licensor.
4.	Example : Wikipedia	Example : Microsoft windows
5.	Sublicensing is allowed.	Sublicensing is not allowed.
6.	No guarantee of further development.	Guarantee of further development.
7.	Fees if any for integration, packing, support and consulting.	Fees are for license, mainteance and upgradation.
8.	Android OS is open source software provided by Google.	An iOS is proprietary software provided by Apple.

1.10 Cloud and Big Data

- The NIST defines cloud computing as : "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models and four deployment models."
- Cloud provider is responsible for the physical infrastructure and the cloud consumer is responsible for application configuration, personalization and data.
- Broad network access refers to resources hosted in a cloud network that are available for access from a wide range of devices. Rapid elasticity is used to describe the capability to provide scalable cloud computing services.
- In measured services, NIST talks about measured service as a setup where cloud systems may control a user or tenant's use of resources by a metering capability somewhere in the system.
- On-demand self-service refers to the service provided by cloud computing vendors that enables the provision of cloud resources on demand whenever they are required.
- The Cloud Cube Model has four dimensions to differentiate cloud formations :
  a) External/Internal b) Proprietary/Open
  c) De-perimeterized / peremeterized d) Outsourced/Insourced.
- **External / Internal :** Physical location of data is defined by external/internal dimension. It defines the organization's boundary.
- **Example :** Information inside a datacenter using a private cloud deployment would be considered internal and data that resided on Amazon EC2 would be considered external.
- **Proprietary / Open :** Ownership is proprietary or open; is a measurement for not only ownership of technology but also its interoperability, use of data and ease of data-transfer and degree of vendor's application's lock-in.
- Proprietary means that the organization providing the service is keeping the means of provision under their ownership. Clouds that are open are using technology that is not proprietary, meaning that there are likely to be more suppliers.
- **De-perimeterized / peremeterized** : Security Ranges : is parameterized or de-parameterized; which measures whether the operations are inside or outside the security boundary, firewall, etc.

- Encryption and key management will be the technology means for providing data confidentiality and integrity in a de-perimeterized model.
- **Outsourced / Insourced : Out-sourcing/In-sourcing;** which defines whether the customer or the service provider provides the service.
- **Outsourced** means the service is provided by a third party. It refers to letting contractors or service providers handle all requests and most of cloud business models fall into this.
- Insourced is the services provided by your own staff under organization control. Insourced means in-house development of clouds.
- Cloud computing is often described as a stack, as a response to the broad range of services built on top of one another under the "**cloud**". A cloud computing stack is a cloud architecture built in layers of one or more cloud-managed services (SaaS, Paas, IaaS, etc.).
- Cloud computing stacks are used for all sorts of applications and systems. They are especially good in microservices and scalable applications, as each tier is dynamically scaling and replaceable.
- The cloud computing pile makes up a threefold system that comprises its lower-level elements. These components function as formalized cloud computing delivery models :
  a) Software as a Service (SaaS)
  b) Platform as a Service (PaaS)
  c) Infrastructure as a Service (IaaS)
- SaaS applications are designed for end-users and delivered over the web.
- PaaS is the set of tools and services designed to make coding and deploying those applications quick and efficient.
- IaaS is the hardware and software that powers it all, including servers, storage, networks and operating systems.

1.10.1 Difference between Cloud Computing and Big Data

Sr. No.	Cloud computing	Big data
1.	It provides resources on demand.	It provides a way to handle huge volumes of data and generate insights.
2.	It refers to internet services from SaaS, PaaS to IaaS.	It refers to data, which can be structured, semi-structured or unstructured.

3.	Cloud is used to store data and information on remote servers.	It is used to describe huge volume of data and information.
4.	Cloud computing is economical as it has low maintenance costs centralized platform no upfront cost and disaster safe implementation.	Big data is highly scalable, robust ecosystem and cost - effective.
5.	Vendors and solution providers of cloud computing are Google, Amazon Web Service, Dell, Microsoft, Apple and IBM.	Vendors and solution providers of big data are Cloudera, Hortonworks, Apache and MapR.
6.	The main focus of cloud computing is to provide computer resources and services with the help of network connection.	Main focus of big data is about solving problems when a huge amount of data generating and processing.

1.10.2 Difference between Cloud Computing and Internet

Sr. No.	Cloud computing	Internet
1.	Cloud computing is a new technology that delivers many types of resources over the Internet.	Internet is a network of networks, which provides software/hardware infrastructure to establish and maintain connectivity of the computers around the world.
2.	Cloud computing allows individuals and businesses to access on-demand computing resources and applications.	The Internet is interconnected with unique identifiers and can exchange data over a network with little or no human interaction.
3.	Cloud computing cannot operate globally without the Internet.	Internet operates without cloud computing.
4.	Cloud computing is owned by a person, company or institution or government.	No single person, company, institution, or government agency controls or owns the Internet.
5.	Cloud computing is an application-based software infrastructure that stores data on remote servers, which can be accessed through the internet.	The Internet provides software/hardware infrastructure to establish and maintain connectivity of the computers.
6.	The Internet is an enabling infrastructure.	Cloud computing is the promise of the utilization of that infrastructure to provide continuous services.

1.11 Mobile Business Intelligence

- Mobile Business Intelligence (BI) or Mobile analytics is the rising software technology that allows users to access information and analytics on their phones

and tablets instead of desktop-based BI systems. Mobile analytics involves measuring and analyzing data generated by mobile platforms and properties, such as mobile sites and mobile applications.

- Analytics is the practice of measuring and analyzing data of users in order to create an understanding of user behavior as well as website or application's performance. If this practice is done on mobile apps and app users, it is called **"mobile analytics"**.
- Mobile analytics is the practice of collecting user behavior data, determining intent from those metrics and taking action to drive retention, engagement and conversion.
- Mobile analytics is similar to web analytics where identification of the unique customer and recording their usages.
- With mobile analytics data, you can improve your cross-channel marketing initiatives, optimize the mobile experience for you customers and grow mobile user engagement and retention.
- Analytics usually comes in the form of a software that integrates into companie's existing websites and apps to capture, store and analyze the data.
- It is always very important for businesses to measure their critical KPIs (Key Performance Indicators), as the old rule is always valid : "If you can't measure it, you can't improve it".
- To be more specific, if a business find out 75 % of their users exit in the shipment screen of their sales funnel, probably there is something wrong with that screen in terms of its design, user interface (UI) or user experience (UX) or there is a technical problem preventing users from completing the process.

Working of Mobile Analytics :

- Most of the analytics tools need a library (an SDK) to be embedded into the mobile app's project code and at minimum an initialization code in order to track the users and screens.
- SDKs differ by platform so a different SDK is required for each platform such as iOS, Android, Windows Phone etc. On top of that, additional code is required for custom event traking.
- With the help of this code, analytics tools track and count each user, app launch, tap, event, app crash or any additional information that the user has, such as device, operating system, version IP address (and probable location).

- Unlike web analytics, mobile analytics tools don't depend on cookies to identify unique users since mobile analytics SDKs can generate a persistent and unique identifier for each device.
- The tracking technology varies between websites, which use either JavaScript or cookies and apps, which use a Software Development Kit (SDF).
- Each time a website or app visitor takes an action, the application fires off data which is recorded in the mobile analytics platform.

1.11.1 Difference between Mobile Analytics and Web Analytics

Sr. No.	Mobile analytics	Web analytics
1.	When web site is using, then mobile user called as USER.	When web site is using, then user called as VISITER.
2.	Interaction with site is called as SESSION.	Interaction with site is called as VISTIS.
3.	On mobile, users have less screen real estate (4 to 7 iches) and interact by toucing, swiping and holding.	On a desktop, users have larger screes (10 to 17 inches) and interact by clicking, double-clicking and using key commands.
4.	Session timeout may be as short as 30 seconds.	Session will end after 30 minutes of inactivity for websites.
5.	Unique users are identified via user IDs.	Cookies are used to identifies user.

1.12 Crowd Sourcing Analytics

- Crowdsourcing is the process of exploring customer's ideas, opinions and thoughts available on the internet from large groups of people aimed at incorporating innovation, implementing new ideas and eliminating product issues.
- Crowdsourcing means the outsourcing of human-intelligence tasks to a large group of unspecified people.via the Internet.
- Crowdsourcing is all about collecting data from users through some services, ideas, or content and then it needs to be stored in a server such that the necessary data can be or provided to users whenever necessary.
- Most users nowadays use Truecaller to find unknown numbers and Google Maps to find out places and the traffic in a region. All the services are based on crowdsourcing.

- Crowdsourced data is a form of secondary data. Secondary data refers to data that is collected by any party other than the researcher. Secondary data provides important context for any investigation into a policy intervention.
- When crowdsourcing data, researchers collect plentiful, valuable and dispersed data at a cost typically lower than that of traditional data collection methods.
- Consider the trade-offs between sample size and sampling issues before deciding to crowdsource data. Ensuring data quality means making sure the platform on which you are collecting crowdsourced data is well-tested.
- Crowdsourcing experiments are normally set up by asking a set of users to perform a task for a very small remuneration on each unit of the task. Amazon Mechanical Turk (AMT) is a popular platform that has a large set of registered remote workers who are hired to perform tasks such as data labeling.
- In data labeling tasks, the crowd workers are randomly assigned a single item in the dataset. A data object may receive multiple labels from different workers and these have to be aggregated to get the overall true label.
- Crowdsourcing allows for many contributors to be recruited in a short period of time, thereby eliminating traditional barriers to data collection. Furthermore, crowdsourcing platforms usually employ their own tools to optimize the annotation process, making it easier to conduct time-intensive labeling tasks. Crowdsourcing data is especially effective in generating complex and free-form labels such as in the case of audio transcription, sentiment analysis, image annotation or translation.
- With crowdsourcing, companies can collect information from customers and use it to their advantage. Brands gather opinions, ask for help, receive feedback to improve their product or service, and drive sales. For instance, Lego conducted a campaign where customers had the chance to develop their designs of toys and submit them.
- To become the winner, the creator had to receive the biggest amount of people's votes. The best design was moved to the production process. Moreover, the winner got a privilege that amounted to a 1 % royalty on the net revenue.
- **Types of Crowdsourcing :** There are four main types of crowdsourcing.
  1. **Wisdom of the crowd :** It is a collective opinion of different individuals gathered in a group. This type is used for decision-making since it allows one to find the best solution for problems.
  2. **Crowd creation :** This type involves a company asking its customers to help with new products. This way, companies get brand new ideas and thoughts that help a business stand out.

3. **Crowd voting :** It is a type of crowdsourcing where customers are allowed to choose a winner. They can vote to decide which of the options is the best for them. This type can be applied to different situations. Consumers can choose one of the options provided by experts or products created by consumers.
4. **Crowdfunding :** It is when people collect money and ask for investments for charities, projects and startups without planning to return the money to the owners. People do it voluntarily. Often, companies gather money to help individuals and families suffering from natural disasters, poverty, social problems, etc.

1.13 Inter and Trans Firewall Analytics

- A firewall is a device designed to control the flow of traffic into and out-of a network. In general, firewalls are installed to prevent attacks. Firewall can be a software program or a hardware device.
- Fig. 1.13.1 shows firewall.

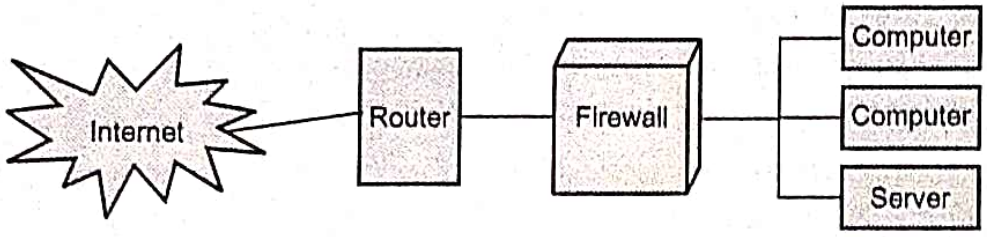


Fig. 1.13.1 Firewall

- Firewalls are software programs or hardware devices that filter the traffic that flows into a user PC or user network through an internet connection. They sift through the data flow and block that which they deem harmful to the user network or computer system.
- Firewalls filter based on IP, UDP and TCP information. Firewall is placed on the link between a network router and Internet or between a user and router. For large organizations with many small networks, the firewall is placed on every connection attached to the Internet.
- Large organizations may use multiple levels of firewall or distributed firewalls, locating a firewall at a single access point to the network.
- Firewalls test all traffic against consistent rules and pass traffic that meets those rules. Many routers support basic firewall functionality. Firewall can also be used to control data traffic.

- Firewall based security depends on the firewall being the only connectivity to the size from outside; there should be no way to bypass the firewall via other gateways; wireless connections.
- Firewall filters out all incoming messages addressed to a particular IP address or a particular TCP port number. It divides a network into a more trusted zone internal to the firewall and a less trusted zone external to the firewall.
- Firewalls may also impose restrictions on outgoing traffic, to prevent certain attacks and to limit losses if an attacker succeeds in getting access inside the firewall.
- Functions of firewall :
  1. **Access control :** Firewall filters incoming as well as outgoing packets.
  2. **Address/Port Translation :** Using network address translation, internal machines, though not visible on the Internet, can establish a connection with external machines on the Internet. NATing is often done by firewall.
  3. **Logging :** Security architecture ensures that each incoming or outgoing packet encounters at least one firewall. The firewall can log all anomalous packets.
- Firewalls can protect the computer and user personal information from :
  1. Hackers who breaks your system security.
  2. Firewall prevents malware and other Internet hacker attacks from reaching your computer in the first place.
  3. Outgoing traffic from your computer created by a virus infection.
- **Firewalls cannot provide protection :**
  1. Against phishing scams and other fraudulent activity
  2. Viruses spread through e-mail
  3. From physical access of your computer or network
  4. For an unprotected wireless network.

Firewall Characteristics

1. All traffic from inside to outside and vice versa, must pass through the firewall.
2. The firewall itself is resistant to penetration.
3. Only authorized traffic, as defined by the local security policy, will be allowed to pass.

1.13.1 Firewall Rules

- The rules and regulations set by the organization. Policy determines the type of internal and external information resources employees can access, the kinds of

programs they may install on their own computers as well as their authority for reserving network resources.

- Policy is typically general and set at a high level within the organization. Policies that contain details generally become too much of a "living document".
- User can create or disable firewall filter rules based on following conditions :
  1. **IP addresses :** System admin can block a certain range of IP addresses.
  2. **Domain names :** Admin can only allow certain specific domain names to access your systems or allow access to only some specific types of domain names or domain name extension.
  3. **Protocol :** A firewall can decide which of the systems can allow or have access to common protocols like IP, SMTP, FTP, UDP, ICMP, Telnet or SNMP.
  4. **Ports :** Blocking or disabling ports of servers that are connected to the internet will help maintain the kind of data flow you want to see it used for and also close down possible entry points for hackers or malignant software.
  5. **Keywords :** Firewalls also can sift through the data flow for a match of the keywords or phrases to block out offensive or unwanted data from flowing in.
- When your computer makes a connection with another computer on the network, several things are exchanged including the source and destination ports. In a standard firewall configuration, most inbound ports are blocked. This would normally cause a problem with return traffic since the source port is randomly assigned. A state is a dynamic rule created by the firewall containing the source-destination port combination, allowing the desired return traffic to pass the firewall.

1.13.2 Types of Firewall

1. Packet filter 2. Application level firewall 3. Circuit level gateway.

- Fig. 1.13.2 shows relation between OSI layer and Firewall.
- Packet filter firewall controls access to packets on the basis of packet source and destination address or specific transport protocol type. It is done at the OSI data link, network and transport layers. Packet filter firewall works on the network layer of the OSI model.
- Packet filters do not see inside a packet; they block or accept packets solely on the basis of the IP addresses and ports. All incoming SMTP and FTP packets are parsed to check whether they should drop or forwarded. But outgoing SMTP and FTP packets have already been screened by the gateway and do not have to be checked by the packet filtering router. Packet filter firewall only checks the header information.

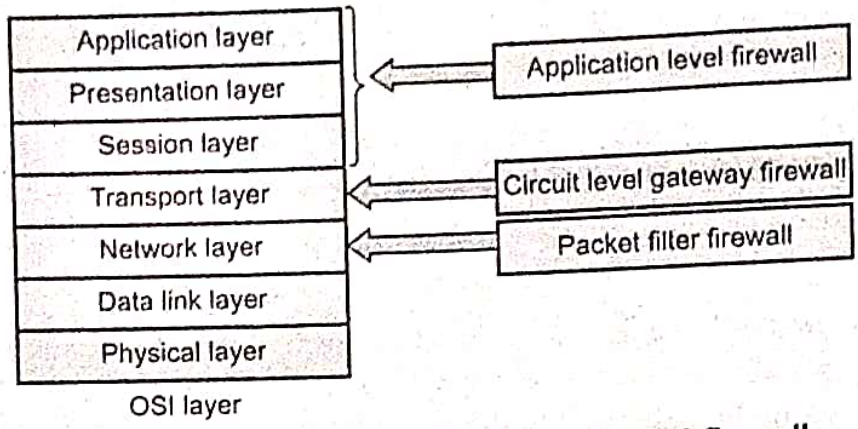


Fig. 1.13.2 Relation between OSI layer and firewall

- Application level gateway is also called a **bastion host.** It operates at the application level. Multiple application gateways can run on the same host but each gateway is a separate server with its own processes.
- These firewalls, also known as **application proxies,** provide the most secure type of data connection because they can examine every layer of the communication, including the application data.
- **Circuit level gateway :** A circuit-level firewall is a second generation firewall that validates TCP and UDP sessions before opening a connection. The firewall does not simply allow or disallow packets but also determines whether the connection between both ends is valid according to configurable rules, then opens a session and permits traffic only from the allowed source and possibly only for a limited period of time.
- It typically performs basic packet filter operations and then adds verification of proper handshaking of TCP and the legitimacy of the session information used in establishing the connection. The decision to accept or reject a packet is based upon examining the packet's IP header and TCP header.
- Circuit level gateway cannot examine the data content of the packets it relays between a trusted network and an untrusted network.

1.13.3 Comparison between Packet Filter and Application Level Gateways

Sr. No.	Packet filter	Application level
1.	Simplest.	Even more complex.
2.	Screens based on connection rules.	Screens based on behavior or proxies.
3.	Auditing is difficult.	Activity can audit.

4.	Low impact on network performance.	High impact on network performance.
5.	Network topology can not hide.	Network topology can hide from attacker.
6.	Transparent to user.	Not transparent to user.
7.	See only addresses and service protocol type.	Sees full data portion of packet.

1.14 Two Marks Questions with Answers

Q.1 What is data science ?

Ans. : Data science is an interdisciplinary field that seeks to extract knowledge or insights from various forms of data. At its core, data science aims to discover and extract actionable knowledge from data that can be used to make sound business decisions and predictions. Data science uses advanced analytical theory and various methods such as time series analysis for predicting the future.

Q.2 What is data ?

Ans. : Data set is a collection of related records or information. The information may be on some entity or some subject area.

Q.3 Define structured data.

Ans. : Structured data is arranged in rows and column format. It helps for applications to retrieve and process data easily. Database management system is used for storing structured data. The term structured data refers to data that is identifiable because it is organized in a structure.

Q.4 What is unstructured data ?

Ans. : Unstructured data is data that does not follow a specified format. Row and columns are not used for unstructured data. Therefore it is difficult to retrieve required information. Unstructured data has no identifiable structure.

Q.5 What is machine-generated data ?

Ans. : Machine-generated data is an information that is created without human interaction as a result of a computer process or application activity. This means that data entered manually by an end-user is not recognized to be machine-generated.

Q.6 Define streaming data.

Ans. : Streaming data is data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously and in small sizes (order of kilobytes).

Q.7 Explain NIST definition of define cloud computing.

Ans. : The NIST defines cloud computing as : Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models and four deployment models.

Q.8 How is big data used in marketing ?

Ans. : Big data offers a snapshot of a business user base, as trends, preferences and behavior patterns can be revealed through data analysis. Companies use these data insights to drive marketing strategies and product decisions.

Q.9 What is a web log file ?

Ans. : Web log file is a log file automatically created and maintained by a web server. Every "hit" to the Web site, including each view of a HTML document, image or other object, is logged. The raw web log file format is essentially one line of text for each hit to the web site. This contains information about who was visiting the site, where they came from and exactly what they were doing on the web site.

Q.10 What are web browsers ?

Ans. : A web browser is a program. Web browser is used to communicate with web servers on the Internet, which enables it to download and display the web pages. Netscape Navigator and Microsoft Internet Explorer are the most popular browser softwares available in the market.

Q.11 What is a web crawler ?

Ans. : A web crawler (also known as a web spider) is a program which browses the World Wide Web in a methodical, automated manner. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches.

Q.12 Define focused crawler.

Ans. : A focused crawler or topical crawler is a web crawler that attempts to download only web pages that are relevant to a pre-defined topic or set of topics.

Q.13 Define a firewall.

Ans. : A firewall is a device designed to control the flow of traffic into and out-of a network. In general, firewalls are installed to prevent attacks. Firewall can be a software program or a hardware device.

Q.14 What are the characteristics of a firewall ?

Ans. : 1. All traffic from inside to outside and vice versa, must pass through the firewall.

2. The firewall itself is resistant to penetration
3. Only authorized traffic, as defined by the local security policy, will be allowed to pass.

Q.15 What is Hadoop ?

Ans. : Hadoop is an open source framework from Apache and is used to store, process and analyze data which are very huge in volume. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

Q.16 Why Hadoop is important ?

Ans. : Hadoop is capable of storing and processing large amounts of data of various kinds. There is no need to preprocess the data before storing it. Hadoop is highly scalable as it can store and distribute large data sets over several machines running in parallel.

UNIT II

2 NoSQL Data Management

Syllabus

Introduction to NoSQL - aggregate data models - key-value and document data models - relationships - graph databases - schemaless databases - materialized views - distribution models - master-slave replication - consistency - Cassandra - Cassandra data model – Cassandra examples - Cassandra clients

Contents

2.1 Introduction to NoSQL

- NoSQL means Not Only SQL, it solves the problem of handling huge volume of data that relational databases cannot handle. NoSQL databases are schema free and are non-relational databases. Most of the NoSQL databases are open source.
- NoSQL is also type of distributed database, which means that information is copied and stored on various servers, which can be remote or local. This ensures availability and reliability of data. If some of the data goes offline, the rest of the database can continue to run.
- NoSQL encompasses structured data, semi-structured data, unstructured data and polymorphic data.
- No SQL database provides a mechanism for storage and retrieval of data that employs less constrained consistency models than traditional relational databases.
- NoSQL is a response to nowadays business data related factors :
  1. Volume and velocity, referring to the ability to handle large datasets that arrive quickly;
  2. Variability, referring to how diverse data types don't fit into structured tables;
  3. Agility, referring to how fast an organization responds to business changes.
- NoSQL databases are very often referred to as data stores rather than data-bases.
- NoSQL systems work on multiple processors and can run on low-cost separate computer systems - No need for expensive nodes to get high-speed performance It supports linear scalability. Every time we add more processors, we get a consistent increase in performance.
- History of NoSQL :
  - The acronym NoSQL was first used in 1998 by Carlo Strozzi while naming his lightweight, open-source "relational" database that did not use SQL. The name came up again in 2009 when Eric Evans and Johan Oskarsson used it to describe non-relational databases.
  - Relational databases are often referred to as SQL systems. The term NoSQL can mean either "No SQL systems" or the more commonly accepted translation of "Not only SQL," to emphasize the fact some systems might support SQL-like query languages.
  - NoSQL developed at least in the beginning as a response to web data, the need for processing unstructured data and the need for faster processing The NoSQL model uses a distributed database system, meaning a system with multiple computers.

- Not only can NoSQL systems handle both structured and unstructured data, but they can also process unstructured big data quickly. This led to organizations such as Facebook, Twitter, LinkedIn, and Google adopting NoSQL systems. These organizations process tremendous amounts of unstructured data, coordinating it to find patterns and gain business insights. Big data became an official term in 2005.

Why NoSQL ?

- It can handle large volumes of structured, semi-structured and unstructured data.
  - Agile sprints, quick iteration and frequent code pushes.
  - Object-oriented programming that is easy to use and flexible.
  - Scale-out architecture.

Types of NoSQL Stores :

1. Column Oriented (Accumulo, Cassandra, Hbase)
2. Document Oriented (MongoDB, Couchbase, Clusterpoint)
3. Key-value (Dynamo, MemcacheDB, Riak)
4. Graph (Allegro, Neo4j, OrientDB)

- NoSQL databases are guaranteed to adhere to two of the CAP properties. Such databases are of several types.
  1. **Key-value store :** Stores in the form of a hash table {Example - Riak, Amazon S3 (Dynamo), Redis}
  2. **Document-based store :** Stores objects, mostly JSON, which is web friendly or supports ODM (Object Document Mappings). {Example - CouchDB, MongoDB}
  3. **Column-based store :** Each storage block contains data from only one column {Example - HBase, Cassandra}
  4. **Graph-based :** Graph representation of relationships, mostly used by social networks. {Example - Neo4J}

2.1.1 The Definition of Four Types of NoSQL Databases

- NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. NoSQL is often interpreted as Not-only-SQL to emphasize that they may also support SQL-like query languages. Most NoSQL databases are designed to store large quantities of data in a fault-tolerant way.
- NoSQL is simply the term that is used to describe a family of databases that are all non-relational. While the technologies, data types and use cases vary widely amount them, it is generally agreed that there are four types of NoSQL databases :

- NoSQL databases can manage information using any of four primary data models : Key-value store, document-based, column based and graph based.

2.1.2 Example and Advantages

- Examples of NoSQL databases
  a) Apache CouchDB, an open source, JSON document-based database that uses JavaScript as its query language.
  b) Elasticsearch, a document-based database that includes a full-text search engine.
  c) Couchbase, a key-value and document database that empowers developers to build responsive and flexible applications for cloud, mobile and edge computing.
- **Advantages**
  a) NoSQL databases have a simple and flexible structure. They are schema-free.
  b) NoSQL databases are based on key-value pairs.
  c) Some store types of NoSQL databases include column store, document store, key value store, graph store, object store, XML store and other data store modes.
  d) Each value in the database has a key. Some NoSQL database stores also allow developers to store serialized objects into the database, not just simple string values.
  e) Open-source NoSQL databases do not require expensive licensing fees and can run on inexpensive hardware, rendering their deployment cost-effective.
- **Disadvantages :**
  a) Most NoSQL databases do not support reliability features that are natively supported by relational database systems.
  b) In order to support reliability and consistency features, developers must implement their own proprietary code, which adds more complexity to the system.

2.1.3 CAP Theorem

- Fig. 2.1.1 shows the three properties of the CAP theorem.
- The theorem states that distributed data systems will offer a trade-off between consistency, availability and partition tolerance. And, that any database can only guarantee two of the three properties :
- **Consistency :** Every node in the cluster responds with the most recent data, even if the system must block the request until all replicas update. If you query a

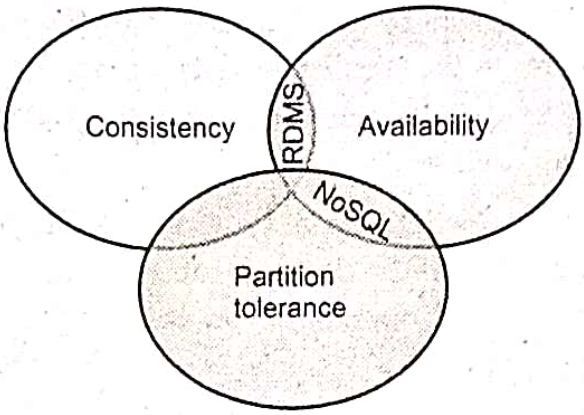


Fig. 2.1.1 CAP theorem

"consistent system" for an item that is currently updating, you will wait for that response until all replicas successfully update. However, you will receive the most current data.

- **Availability :** Every node returns an immediate response, even if that response is not the most recent data. If you query an "available system" for an item that is updating, you will get the best possible answer the service can provide at that moment.
- **Partition tolerance :** Guarantees the system continues to operate even if a replicated data node fails or loses connectivity with other replicated data nodes.

2.1.4 Comparison of SQL and NoSQL Databases

Sr. No.	SQL	NoSQL
1.	SQL databases are relational.	NoSQL databases are non-relational.
2.	SQL databases are vertically scalable.	NoSQL databases are horizontally scalable.
3.	SQL databases use structured query language and have a predefined schema.	NoSQL databases have dynamic schemas for unstructured data.
4.	SQL databases are table-based.	NoSQL databases are document, key-value, graph, or wide-column stores.
5.	SQL databases are better for multi-row transactions.	while NoSQL is better for unstructured data like documents or JSON.

2.2 Aggregate Data Models

- Aggregate means a collection of objects that are treated as a unit. In NoSQL Databases, an aggregate is a collection of data that interact as a unit. Moreover, these units of data or aggregates of data form the boundaries for the ACID operations.

- Aggregate data models in NoSQL make it easier for the databases to manage data storage over the clusters as the aggregate data or unit can now reside on any of the machines. Whenever data is retrieved from the database all the data comes along with the aggregate data models in NoSQL.
- Aggregate data models in NoSQL do not support ACID transactions and sacrifice one of the ACID properties. With the help of aggregate data models in NoSQL, we can easily perform OLAP operations on the database. We can achieve high efficiency of the aggregate data models in the NoSQL database if the data transactions and interactions take place within the same aggregate.

2.2.1 Key-value Store

- In the key-value structure, the key is usually a simple string of characters and the value is a series of uninterrupted bytes that are opaque to the database. Key-value store is like a relational database with only two columns : The key or attribute name and the value.
- Fig. 2.2.1 shows key-value store.

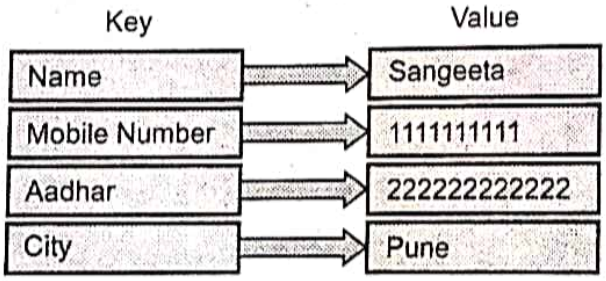


Fig. 2.2.1 Key-value store

- Saves data as a group of key value pairs, which are made up of two data items that are linked. The link between the items is a "key" which acts as an identifier for an item within the data and the "value" that is the data that has been identified.
- The data itself is usually some primitive data type (string, integer, array) or a more complex object that an application needs to persist and access directly.
- This replaces the rigidity of relational schemas with a more flexible data model that allows developers to easily modify fields and object structures as their applications evolve.
- Key value systems treat the data as a single opaque collection which may have different fields for every record.
- In each key value pair,
  a) The key is represented by an arbitrary string
  b) The value can be any kind of data like an image, file, text or document.

- In general, key-value stores have no query language. They simply provide a way to store, retrieve, and update data using simple GET, PUT and DELETE commands.
- The simplicity of this model makes a key-value store fast, easy to use, scalable, portable and flexible.
- **Advantages of key value stores :**
  a) The secret to its speed lies in its simplicity. The path to retrieve data is a direct request to the object in memory or on disk.
  b) The relationship between data does not have to be calculated by a query language, there is no optimization performed.
  c) They can exist on distributed systems and do not need to worry about where to store indexes.
- **Disadvantages of key value stores :**
  a) No complex query filters
  b) All joins must be done in code
  c) No foreign key constraints
  d) No trigger.

2.2.2 Document-based

- A document is an object and keys (strings) that have values of recognizable types, including numbers, Booleans and strings, as well as nested arrays and dictionaries. All data is stored in one table, so there is no need for cross-referencing and instead of storing information in a table, it is stored in a document.
- Fig. 2.2.2 shows document based data model.

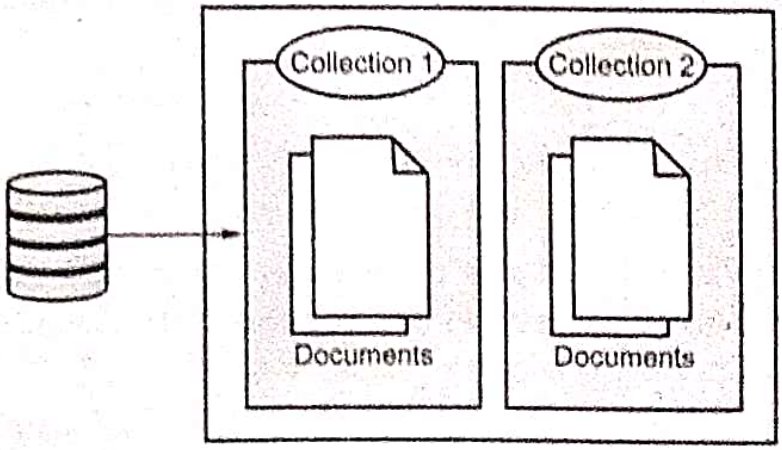


Fig. 2.2.2 Document based data model

- Document databases are designed for flexibility. They are not typically forced to have a schema and are therefore easy to modify.

- If an application requires the ability to store varying attributes along with large amounts of data, document databases are a good option.
- Document stores work with multiple formats including XML and JSON. This allows for storage and retrieval of data without an impedance match.
- Terminologies in document data store are as follows :
  a) A table is called a **collection**
  b) A row is called a **document**
  c) A column in called a **field.**
- Typical use cases for document stores include the storage and retrieval of catalogs, blog posts, news articles and data analysis.
- MongoDB and Apache CouchDB are examples of popular document-based databases.
- Do not use document databases for transactions across multiple documents (records) and Ad hoc cross-document queries.
- **Advantages of document based model :**
  a) Faster retrieval of data.
  b) Dynamic architecture for unstructured data and storage options
  c) Sharing for horizontal scalability
  d) Replication is managed internally, so chances of accidental loss of data is negligible.
- **Disadvantages of document data model :**
  a) No views, triggers, scripts or stored procedure.
  b) Relationship not well defined.
  c) No support for transactions, which could lead to data corruption.

2.2.3 Column-based

- Column-based is also called **'wide column'** models enabling very quick data access using a row key, column name and cell timestamp.
- The flexible schema of these types of databases means that the columns do not have to be consistent across records and you can add a column to specific rows without having to add them to every single record.
- It is also called a **two-level map** as it offers a two-level aggregate structure.
- As data is organized into columns, we have better indexing compared to other key-value stores. Also, when it comes to updates, multiple column block updates can be aggregated.

- Column store databases were born when Google open sourced its implementation of a Column store NoSQL database called **Big Table**. Apparently, the data for the well-known Google e-mail service, Gmail, is stored in the Google Big Table NoSQL Database.
- The wide, columnar stores data model, like that found in Apache Cassandra, are derived from Google's BigTable paper.
- Organizations mostly use Column data stores for data warehousing and data processing, which is evident in services such as Amazon Redshift.
- **Advantages of column data stores :**
  a) Column stores are very efficient at data compression and/or partitioning.
  b) Columnar stores can be loaded extremely fast.
  c) Columnar databases are very scalable.
  d) Due to their structure, columnar databases perform particularly well with aggregation queries.
- **Disadvantages of column data store :**
  a) Updates can be inefficient. The fact that columnar families group attributes, as opposed to rows of tuples, works against it.
  b) If multiple attributes are touched by a join or query, this may also lead to column storage experiencing slower performance.
  c) It is also slower when deleting rows from columnar systems, as a record needs to be deleted from each of the record files.

2.2.4 Graph-based

- The modern graph database is a data storage and processing engine that makes the persistence and exploration of data and relationships more efficient.
- Graph-based data models store data in nodes that are connected by edges. These Aggregate Data Models in NoSQL are widely used for storing the huge volumes of complex aggregates and multidimensional data having many interconnections between them.
- In graph theory, structures are composed of vertices and edges, or what would later be called **"data relationships"**.
- Graphs behave similarly to how people think, in specific relationships between discrete units of data. This database type is particularly useful for visualizing, analyzing, or helping to find connections between different pieces of data.

- As a result, businesses leverage graph technologies for recommendation engines, fraud analytics and network analysis. Examples of graph-based NoSQL databases include Neo4j and JanusGraph.
- Graph databases can be used to analyze customer interactions, social media and scientific applications where it is crucial to traverse long relationship graphs to better understand data.
- **Advantages of graph data :**
  a) More descriptive queries
  b) Greater flexibility in adapting your model
  c) Greater performance when traversing data relationships.
- **Disadvantages of graph data stores :**
  a) Difficult to scale
  b) No standard language.

2.2.5 NoSQL Key/Value Database : MongoDB

- MongoDB is an open-source document database that provides high performance, high availability and automatic scaling. MongoDB is one of the most popular open-source NoSQL databases written in C++. As of February 2015, MongoDB is the fourth most popular database management system. It was developed by a company 10gen which is now known as **MongoDB Inc.**
- Why use MongoDB ?
  a) Simple queries
  b) Functionality provided applicable to most web applications
  c) Easy and fast integration of data
  d) No ERD diagram
  e) Not well suited for heavy and complex transactions systems.
- MongoDB did not provides any command to create a "database". Actually, you do not need to create it manually, because, MangoDB will create it on the fly, during the first time you save the value into the defined collection (or table in SQL) and database.
- MongoDB is a document-oriented database which stores data in JSON-like documents with dynamic schema. It means you can store your records without worrying about the data structure such as the number of fields or types of fields to store values. MongoDB documents are similar to JSON objects.
- MongoDB stores data records as BSON documents. BSON is a binary representation of JSONdocuments, though it contains more data types than JSON.

- MongoDB stores data in documents in-spite of tables. You can change the structure of records simply by adding new fields or deleting existing ones. This ability of MongoDB helps you to represent hierarchical relationships, to store arrays and other more complex structures easily.
- MongoDB uses Mongo server and Mongo shell commands to fetch records or the information from the database (i.e. collections). Few areas where MongoDB is ideal are big data, user data management, mobile and social infrastructure, content management and delivery, data hub.
- A MongoDB instance may have zero or more databases. A database may have zero or more 'collections'. A collection may have zero or more 'documents'. A document may have one or more 'fields'. MongoDB 'Indexes' function much like their RDBMS counterparts.
- Database is a physical container for collections. Collection is a group of documents and is similar to an RDBMS table. A document is a set of key-value pairs. Documents have dynamic schema.
- MongoDB documents are composed of field-and-value pairs and have the following structure :

```
{
  field1: value1,
  field2: value2,
  field3: value3,
  ...
  fieldN: valueN
}
```

- The value of a field can be any of the BSON data types, including other documents, arrays and arrays of documents. MongoDB supports many data types such as : String, integer, boolean, double, arrays, timestamp, object, null, symbol, date, code and binary data.
- Fig. 2.2.3 shows relation between SQL terms and MangoBD terms.

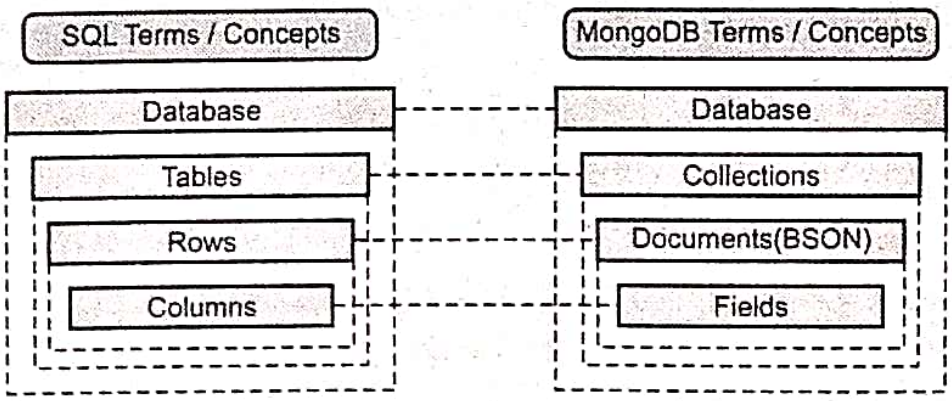


Fig. 2.2.3 Relation between SQL terms and MangoBD terms

- MongoDB uses MongoDB query language and supports Ad hoc queries replication and sharding. Sharding is a feature of MongoDB that helps it to operate as a distributed data system.
- Sharding is used by MongoDB to store data across multiple machines. It uses horizontal scaling to add more machines to distribute data and operation with respect to the growth of load and demand.
- Sharding arrangement in MongoDB has mainly three components :

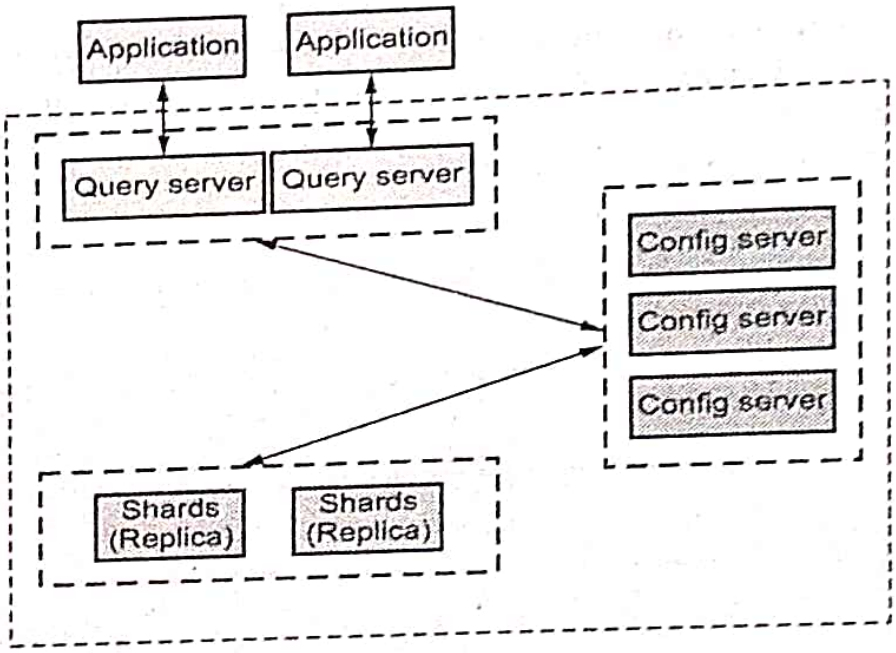


Fig. 2.2.4 Sharding by MongoDB

a) **Shards or replica sets :** Each shard serves as a separate replica set. They store all the data. They target to increase the consistency and availability of the data.

b) **Configuration servers :** They are like the managers of the clusters. These servers contain the cluster's metadata. They actually have the mapping of the cluster's data to the shards. When a query comes, query routers use these mappings from the config servers to target the required shard.

c) **Query router :** The query router is mongo instances which serve as interfaces for user applications. They take in the user queries from the applications and serve the applications with the required results.

Advantages of MangoDB :

- MongoDB is a schema - less document type database.
- MongoDB supports field, range based query, regular expression for searching the data from the stored data.

- MongoDB is very easy to scale up or down.
- It uses internal memory for storing the working temporary datasets for which it is much faster.
- MongoDB support primary and secondary indexes on any field.
- MongoDB supports replication of databases.
- MongoDB can be used as a file storage system which is known as a **GridFS**.

2.3 Schemaless Databases

- Since NoSQL does not require a schema, there is no blueprint on how data should be stored and therefore varies between databases. Generally, there are two ways that NoSQL data storage functions :
  1. On-the-disk using B-Trees, with the top of it being permanently in RAM.
  2. In-memory where it is all on RAM using RB-Trees and anything stored on the disc is just an append.
- Schemaless databases are a type of NoSQL databases that do not have a predefined schema or structure for data. This means that data can be inserted and retrieved without adhering to a specific structure and the database can adapt to changes in data over time without requiring schema migrations or changes.
- Schemaless database manages information without the need for a blueprint. The onset of building a schemaless database does not rely on conforming to certain fields, tables, or data model structures.
- There is no Relational Database Management System (RDBMS) to enforce any specific kind of structure. In other words, it is a non-relational database that can handle any database type, whether that be a key-value store, document store, in-memory, column-oriented, or graph data model.
- In actuality, there is no such thing as schema-less dataset :
  1. In a relational database, the schema is explicit and created separately in advance.
  2. In column-based databases, we create a fresh schema for each row and in fact, we often reuse schema fragments from rows that are grouped together. The same is true for document databases.
  3. In column-based and also in document databases, users directly query data based on the schema.
  4. In graph-based databases, we are in essence building the schema as we build the data.

- In schemaless databases, information is stored in JSON-style documents which can have varying sets of fields with different data types for each field. So, a collection could look like this :

```
{
   name : "Joe", age : 30, interests : 'football' }
{
   name : "Kate", age : 25
   }
```

- In the above condition, the data itself normally has a fairly consistent structure. With the schemaless MongoDB database, there is some additional structure, the system namespace contains an explicit list of collections and indexes. Collections may be implicitly or explicitly created, indexes must be explicitly declared.
- **Benefits of using schemaless databases :**
  1. Flexibility : Schemaless databases allow for greater flexibility in data modeling.
  2. Scalability : Schemaless databases are designed for scalability, as they can handle large amounts of unstructured data with ease.
  3. Reduced complexity : Schemaless databases can reduce the complexity of data modeling and development.
  4. Good support for non-uniform data.
- **Disadvantages :**
  1. Potentially inconsistent names and data types for a single value.
  2. Management of the implicit schema migrates into the application layer.

2.4 Materialized Views

- Materialized views solve the problem of views. The views provide a mechanism to hide from the client whether data is derived data or base data. Views are used when data is to be accessed infrequently and the data in a table gets updated on a frequent basis.
- A materialized view is a replica of a target master from a single point in time. The master can be either a master table at a master site or a master materialized view at a materialized view site. A materialized view is like a cache, a copy of the data that can be accessed quickly.
- If a regular view is a saved query, a materialized view is a saved query plus its results stored as a table.
- NoSQL databases do not have views, they may have precomputed and cached queries and they reuse the term "materialized view" to describe them NoSql.

- We can use materialized views to achieve one or more of the following goals :
  1. Ease network loads
  2. Create a mass deployment environment
  3. Enable data subsetting
  4. Enable disconnected computing.
- Two methods are used for building a materialized view :
  1. Eager approach : user update the materialized view at the same time update the base data for it. In this case, adding an order would also update the purchase history aggregates for each product. This method is used when more frequent reads of the materialized view than writes.
  2. The application database approach is valuable here as it makes it easier to ensure that any updates to base data also update materialized views.
- Materialized views can be built outside of the database by reading the data, computing the view and saving it back to the database.

2.5 Distribution Models

- Ability of NoSql is to run a database on a large cluster. As data volumes increase, it becomes more difficult and expensive to scale up, so it is necessary to buy a bigger server to run the database on.

2.5.1 Single Server

- Database is run on a single machine which handles all the reads and writes to the data store. Organizations prefer a single server because it eliminates all the complexities that the other options introduce.
- Single server is easy to manage for application developers. Lot of NoSQL databases are designed around the idea of running on a cluster, it can make sense to use NoSQL with a single-server distribution model if the data model of the NoSQL store is more suited to the application.
- Single-server configuration is suitable for graph-database.
- If data usage is mostly about processing aggregates, then a single-server document or key-value store may be useful.

2.5.2 Sharding

- Sharding is a method for distributing a single dataset across multiple databases, which can then be stored on multiple machines. This allows for larger datasets to be split into smaller chunks and stored in multiple data nodes, increasing the total storage capacity of the system.

- Sharding is a form of scaling known as **horizontal scaling** or **scale-out**, as additional nodes are brought on to share the load. Horizontal scaling allows for near-limitless scalability to handle big data and intense workloads.
- Sharding is also known as **data partitioning**. Many NoSQL databases offer auto-sharding. Fig. 2.5.1 shows Sharding.

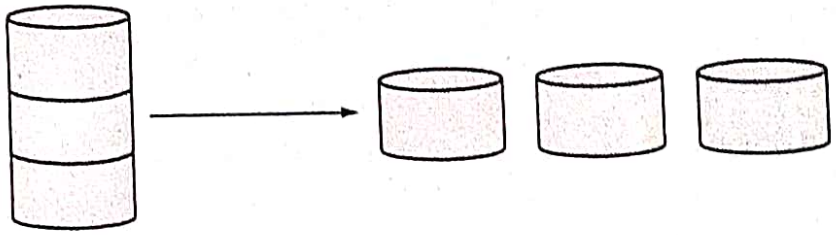


Fig. 2.5.1 Sharding

- Sharding is the process of splitting a large dataset into many small partitions which are placed on different machines. Each partition is known as a "**shard**".
- Each shard has the same database schema as the original database. Most data is distributed such that each row appears in exactly one shard. The combined data from all shards is the same as the data from the original database. The load is balanced out nicely between servers, for example, if we have five servers, each one only has to handle 20 % of the load.
- The NoSQL framework is natively designed to support automatic distribution of the data across multiple servers including the query load. Both data and query replacements are automatically distributed across multiple servers located in the different geographic regions and this facilitates rapid, automatic and transparent replacement of the data or query instances without any disruption.
- Sharding is particularly valuable for performance because it can improve both read and write performance. Using replication, particularly with caching, can greatly improve read performance but does little for applications that have a lot of writes.
- **Advantages of Sharding**
  a) **Faster performance :** There are more servers available to handle input/output.
  b) **Horizontal scaling :** We can quickly add additional servers to a cluster.
  c) **Costs :** Horizontal scaling can often be less expensive than vertical scaling.
  d) **Distribution/uptime :** A horizontally scaled distributed database can achieve better uptime than a traditional single server.

- **Disadvantages of Sharding**
  a) Complexity : Depending on the database system, sharding complexity can vary.
  b) Rebalancing : When adding additional machines to a cluster, the shards will likely need to be rebalanced to distribute data evenly.
  c) Increased infrastructure costs.

2.5.3 Master-slave Replication

- We replicate data across multiple nodes. One node is designed as primary (master), others as secondary (slaves). Master is responsible for processing any updates to that data. A replication process synchronizes the slaves with the master.
- Master is the authoritative source for the data. It is responsible for processing any updates to that data. Masters can be appointed manually or automatically.
- Slaves is a replication process that synchronizes the slaves with the master. After a failure of the master, a slave can be appointed as new master very quickly. Fig. 2.5.2 shows master-slave replication.

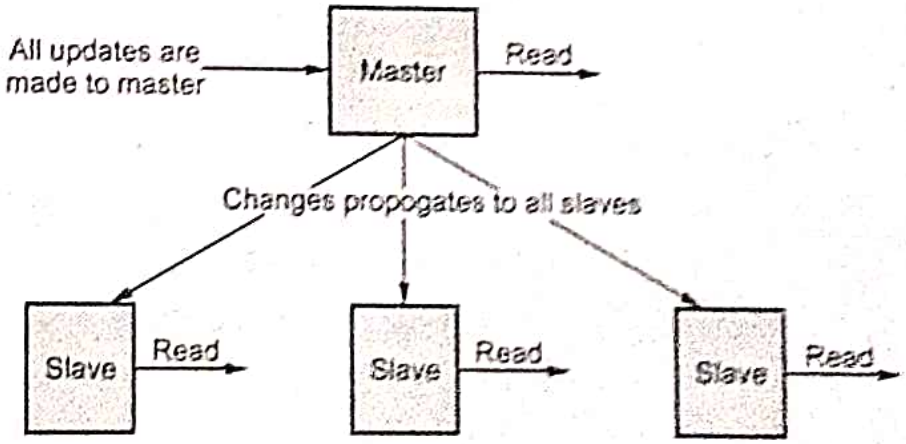


Fig. 2.5.2 Master-slave replication

- Master-slave replication is most helpful for scaling when we have a read-intensive dataset. It will scale horizontally to handle more reads.
- This design offers **read resilience**. Even if one or more of the servers fails, the remaining servers can keep offering read access. This can help a lot with read-heavy applications, but will offer little benefit to write-intensive applications.
- As the slaves are exact replicas of the master server, one of them can assume the role of the master in case the master fails. In fact most of the time you can simply create a set of nodes and have them automatically decide who would be the master. There are some consistency issues that occur due to the delay in updating between master and slaves.

- Masters can be appointed manually or automatically. In manual appointing performed when we configure our cluster and we configure one node as the master. With automatic appointment, we create a cluster of nodes and they elect one of themselves to be the master.
- Problems of master-slave replication :
  1. Does not help with scalability of writes
  2. Provides resilience against failure of a slave, but not of a master
  3. The master is still a bottleneck.

2.5.4 Peer-to-Peer Replication

- In a peer-to-peer replication setup the various nodes are all "equals". Any node can accept reads as well as writes and they communicate these writes to each other. In peer-to-peer replication updates on any one server are replicated to all other associated servers.
- Fig. 2.5.3 shows peer-to-peer replications.

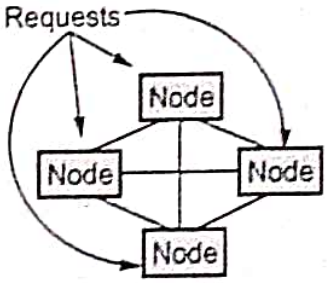


Fig. 2.5.3 Peer-to-peer replications

- The advantage of this setup is its read and write resilience. One node's failure does not cause problems, as the remaining nodes can continue their work without losing a beat.
- The problem that arises is that of consistency. For example we may have conflicting write requests that come to different nodes and then those nodes attempt to communicate those requests to the rest of the nodes. This could lead to considerable inconsistencies.
- There are various ways to resolve this problem. The most standard approach would be to have the replicas communicate their writes first before they "accept" them. Once a majority of the replicas has confirmed a write, it can now be considered as having been successfully performed and a response sent to the client. This requires a certain amount of network traffic in coordinating these writes.

- There is a problem of write-write conflict. Two users can update different copies of the same record stored on different nodes at the same time is called a **write-write conflict.**

2.5.5 Combining Sharding and Replication

- Sharding and replication can be combined to get a better response. If we use both master-slave replication With Sharding and Peer-to-peer replication with Sharding.

1. Master-slave replication and Sharding :

- We have multiple masters, but each data item only has a single master.
- A node can be a master for some data and a slave for others.

2. Peer-to-peer replication and Sharding :

- A common strategy for column-family databases.
- A good starting point for peer-to-peer replication is to have a replication factor of 3, so each shard is present on three nodes.

2.5.6 Difference between Replication and Sharding

Sr. No.	Replication	Sharding
1.	The primary server node copies data onto secondary server nodes. This can help increase data availability and act as a backup, in case the primary server fails.	Sharding : Handles horizontal scaling across servers using a shard key.
2.	Replication copies data across multiple servers.	Sharding distributes different data across multiple servers.
3.	Each bit of data can be found in multiple places.	Each server acts as the single source for a subset of data.
4.	Replicated servers contain identical copies of the entire database.	Sharded database servers each contain a part of the overall data, i.e. they store different data on separate nodes.
5.	More read requests.	It can improve both reads and writes.

2.6 Consistency

- The CAP theorem is important when considering a distributed database, since we must make a decision about what we are willing to give up. The database we choose will lose either availability or consistency.. Reading about NoSQL databases we can face the concept of quorum. A quorum is the minimal number of nodes that must respond to a read or write operation to be considered complete.

Of course having a maximum quorum and querying all servers is the way we can determine the correct result.

- *Consistency can be simply defined by how the copies from the same data may vary within the same replicated database system.*
- Nowadays systems need to scale. The "traditional" monolithic database architecture, based on a powerful server, does not guarantee the high availability and network partition required by today's web-scale systems, as demonstrated by the CAP theorem. To achieve such requirements, systems cannot impose strong consistency.
- In the past, almost all architectures used in database systems were strongly consistent. In these cases, most architectures would have a single database instance only responding to a few hundred clients. Nowadays, many systems are accessed by hundreds of thousands of clients, so there was a mandatory requirement to system's architectures that scale. However, considering the CAP theorem, high-availability and consistency do conflict on distributed systems when subject to a network partition event.

2.6.1 Update Consistency

- Two users updating the same data item at the same time is called **write-write conflict.**
- When the writes reach the server of the two users, the server will serialize them and decide to apply one, then the other. First user's update would be applied and immediately overwritten by the second user.
- In this case first user's is a **lost update**. Here the lost update is not a big problem. We see this as a failure of consistency because second user's update was based on the state before first user's update, yet was applied after it.
- Approaches for maintaining consistency in the case of concurrency are often **described as pessimistic or optimistic.**
- A pessimistic approach works by preventing conflicts from occurring; an optimistic approach lets conflicts occur, but detects them and takes action to sort them out.
- For update conflicts, the most common pessimistic approach is to have write locks, so that in order to change a value we need to acquire a lock and the system ensures that only one client can get a lock at a time.
- So both users would attempt to acquire the write lock, but only the first user would succeed. Second user would then see the result of the first user's write before deciding whether to make his own update.

- A common optimistic approach is a **conditional update** where any client that does an update tests the value just before updating it to see if it is changed since his last read.
- Both the pessimistic and optimistic approaches rely on a consistent serialization of the updates and it is possible for a single server.
- Two general solutions for write-write conflict are as follows :
  1. Pessimistic approach : Preventing conflicts from occurring. Also acquire write locks before update.
  2. Optimistic approach : Lets conflicts occur, but detects them and takes actions to resolve them.
- If there are more than one server i.e. peer-to-peer replication, then two nodes might apply the updates in a different order, resulting in a different value for the telephone number on each peer. Sequential consistency is used in distributed systems.
- Optimistic way to handle a **write-write conflict** is to save both updates and record that they are in conflict. Replication makes it much more likely to run into write-write conflicts. If different nodes have different copies of some data which can be independently updated, then we will get conflicts unless we take specific measures to avoid them. Using a single node as the target for all writes for some data makes it much easier to maintain update consistency.

2.6.2 Read Consistency

- Problem : One user reads in the middle of another user's writing.
- It is called **read-write conflict**, inconsistent reading. This leads to logical inconsistency.
- In NoSQL databases, read consistency refers to the level of consistency between multiple read operations on the same data. In a distributed database, where data can be replicated across multiple nodes, ensuring read consistency can be challenging.
- Aggregate-oriented databases do support atomic updates, but only within a single aggregate. This means that we will have logical consistency within an aggregate but not between aggregates.
- The length of time an inconsistency is present is called the **inconsistency window**. A NoSQL system may have a quite short inconsistency window.
- There are different levels of read consistency available in NoSQL databases, ranging from eventual consistency to strong consistency.

- Eventual consistency allows for a certain degree of inconsistency to occur between different replicas of data. In this model, the database guarantees that all updates will eventually propagate to all nodes, but it makes no guarantees about how long this will take or about the order in which updates will be applied.
- **Read-your-writes consistency** means that once we have updated a record, all of our subsequent reads of that record will return the updated value.
- **Session consistency** means read-your-writes consistency but at session level. Session can be identified with a conversation between a client and a server. As long as the conversation continues, we will read everything we have written during this conversation. If the session ends and we start another session with the same server, there is no guarantee that we can read values we have written during previous conversation.
- Session consistency is of two types : Sticky session and version stamps.

2.6.3 Quorums

- Quorum consistency is used in systems where consistency is more important than availability (CAP theorem) for write and read.
- In systems with multiple replicas there is a possibility that the user reads inconsistent data. This happens say when there are 2 replicas, N1 and N2 in a cluster and a user writes value v1 to node N1 and then another user reads from node N2 which is still behind N1 and thus will not have the value v1, so the second user will not get the consistent state of data.
- In order to achieve a state where at least one node has consistent data we use quorum consistency.
- Fig. 2.6.1 shows write and read quorums.

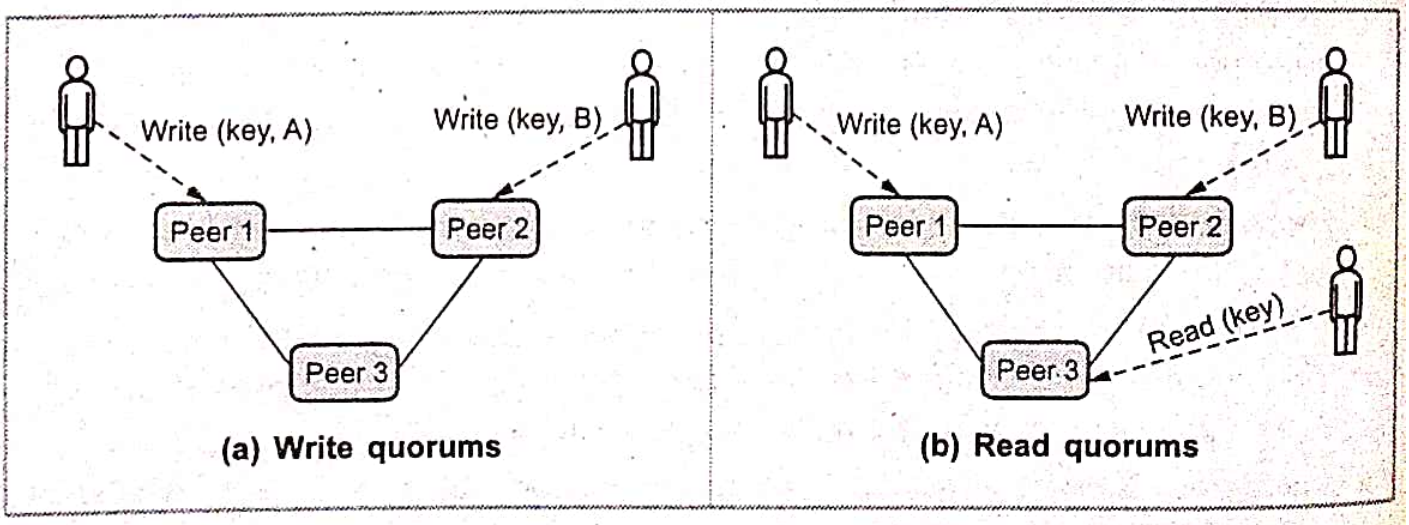


Fig. 2.6.1

- Quorum is achieved when nodes follow the below protocol : $w + r > n$

  where n = Nodes in the quorum group,
  w = Minimum write nodes,
  r = Minimum read nodes

- Here w is our write quorum and r is our read quorum.

2.6.4 Relaxing Durability

- When write is committed, the change is permanent.
- In some cases, strict durability is not essential and it can be traded for scalability (write performance).
- A simple way to relax durability is to store data in memory and flush to disk regularly. If the system shuts down, we lose updates in memory.

2.7 Cassandra

- Cassandra is a column NoSQL database. It was initially developed by Facebook to fulfill the needs of the company's Inbox Search services. In 2009, it became an Apache Project.
- Apache Cassandra is an open source, distributed, NoSQL database. Apache Cassandra is a distributed database system using a shared nothing architecture.
- Apache Cassandra was initially designed at Facebook using a Staged Event-Driven Architecture (SEDA) to implement a combination of Amazon's Dynamo distributed storage and replication techniques and Google's Bigtable data and storage engine model.
- A columnar database, also called a **column-oriented database** or a **wide-column store**, is a database that stores the values of each column together, rather than storing the values of each row together.
- Columnar databases are well suited for big data processing, Business Intelligence (BI) and analytics.
- Cassandra provides tunable consistency i.e. users can determine the consistency level by tuning it via read and write operations. Cassandra enables users to configure the number of replicas in a cluster that must acknowledge a read or write operation before considering the operation successful.
- Cassandra uses a gossip protocol to discover node state for all nodes in a cluster. Cassandra is designed to handle "big data" workloads by distributing data, reads and writes (eventually) across multiple nodes with no single point of failure.

- Features of Cassandra :
  1) **Elastic scalability :** Cassandra is highly scalable; it allows to add more hardware to accommodate more customers and more data as per requirement.
  2) **Always on architecture :** Cassandra has no single point of failure.
  3) **Fast linear-scale performance :** Cassandra is linearly scalable, i.e., it increases throughput as we increase the number of nodes in the cluster.
  4) **Flexible data storage :** Cassandra accommodates all possible data formats including : Structured, semi-structured and unstructured.
  5) **Transaction support :** Cassandra supports properties like ACID.
  6) **Easy data distribution :** Cassandra provides the flexibility to distribute data where you need by replicating data across multiple datacenters.

2.7.1 Cassandra Architecture

- Fig. 2.7.1 shows Cassandra architecture.

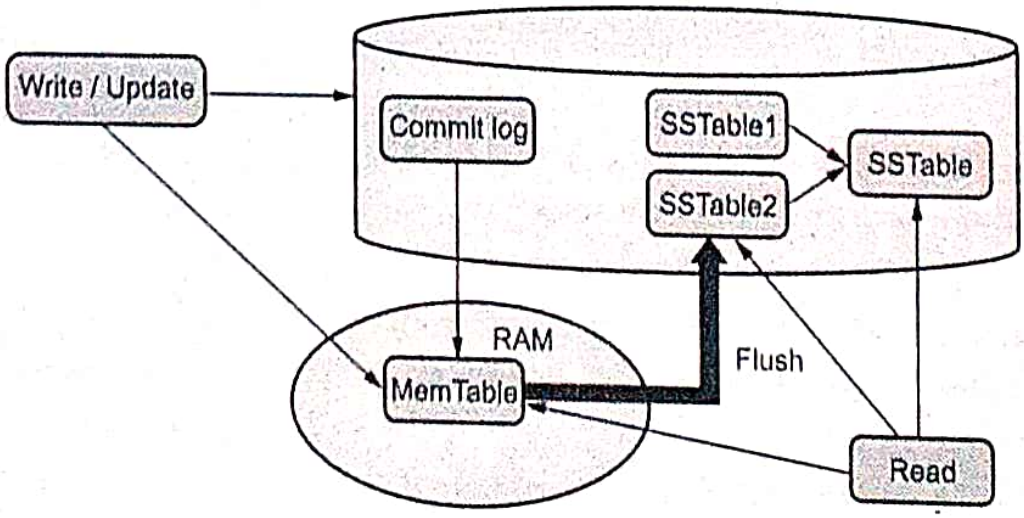


Fig. 2.7.1 Cassandra architecture

- Components of Cassandra architecture are node, data center, cluster, commit log, memtable, SSTable, Bloom Filters and Cassandra query language.
- Node : A Cassandra node is a place where data is stored.
- Data center : Data center is a collection of related nodes.
- Cluster : A cluster is a component which contains one or more data centers.
- Commit log : In Cassandra, the commit log is a crash-recovery mechanism. Every write operation is written to the commit log.
- Mem-table : A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem-tables.

- SSTable : It is a disk file to which the data is flushed from the mem-table when its contents reach a threshold value.
- Bloom filter : Bloom filters are very fast, nondeterministic algorithms for testing whether an element is a member of a set. It is a special kind of cache. Bloom filters are accessed after every query.
- Each node in a Cassandra cluster also maintains a sequential commit log of write activity on disk to ensure data integrity. These writes are indexed and written to an in-memory structure called a **memtable.**
- A memtable can be thought of as a write-back cache where write I/O is directed to cache with its completion immediately confirmed by the host. This has the advantage of low latency and high throughput. The memtable structure is kept in Java heap memory by default.
- SSTables : When the commit log gets full, a flush is triggered and the contents of the memtable are written to disk into an SSTables data file. At the completion of this process the memtable is cleared and the commit log is recycled. Cassandra automatically partitions these writes and replicates them throughout the cluster.

2.7.2 Cassandra Data Model

- Some of the features of Cassandra data model are as follows :
  1) Data in Cassandra is stored as a set of rows that are organized into tables.
  2) Tables are also called **column families.**
  3) Each row is identified by a primary key value.
  4) Data is partitioned by the primary key.
- Data modelling in Cassandra uses a query-driven approach, in which specific queries are the key to organizing the data. The main goal of Cassandra data modeling is to develop and design a high-performance and well-organized Cluster.
- Apache Cassandra data model components include keyspaces, tables and columns :
  a) Cassandra stores data as a set of rows organized into tables or column families
  b) A primary key value identifies each row
  c) The primary key partitions data
  d) We can fetch data in part or in its entirety based on the primary key.
- Cassandra data model provides a mechanism for data storage. The components of Cassandra data model are keyspaces, tables and columns.

- Fig. 2.7.2 shows Cassandra data model

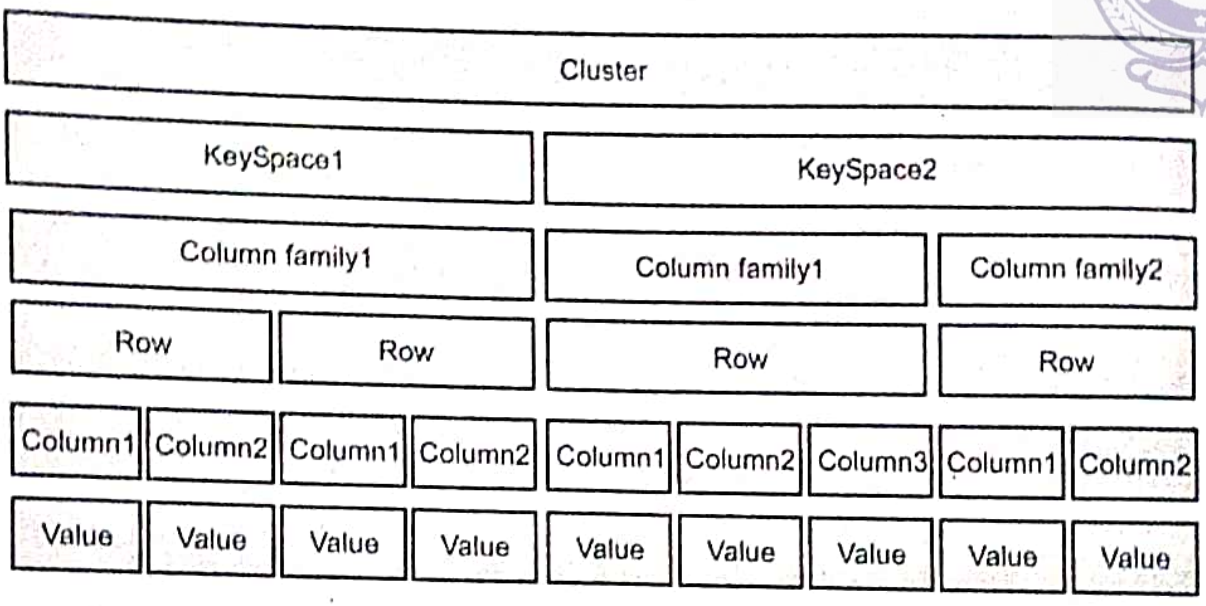


Fig. 2.7.2 Cassandra data model

1. Keyspaces :

- At a high level, the Cassandra NoSQL data model consists of data containers called **keyspaces**. Keyspaces are similar to the schema in a relational database. Typically, there are many tables in a keyspace.
- Features of keyspaces are :
  a) A keyspace needs to be defined before creating tables, as there is no default keyspace.
  b) A keyspace can contain any number of tables and a table belongs only to one keyspace. This represents a one-to-many relationship.
  c) Replication is specified at the keyspace level. For example, replication of three implies that each data row in the keyspace will have three copies.

2. Tables :

- Tables, also called **column families** in earlier iterations of Cassandra, are defined within the keyspaces. Tables store data in a set of rows and contain a primary key and a set of columns.
- Cassandra tables are used to hold the actual data in the form of rows and columns. A table in Cassandra must be created with the primary key during table creation time, post that it can not be altered.
- To alter the table new tables should be created with existing data. The primary key would be used to locate and order the data.

- Some of the features of tables are :
  a) Tables have multiple rows and columns. As mentioned earlier, a table is also called **column family** in the earlier versions of Cassandra.
  b) It is still referred to as column family in some of the error messages and documents of Cassandra.
  c) It is important to define a primary key for a table.

3. **Columns :**
- Columns define data structure within a table. There are various types of columns, such as Boolean, double, integer and text.
- Cassandra column is used to store a single piece of data. The column can consist of various types of data such as big integer, double, text, float and Boolean.
- Each column value has a timestamp associated with it that shows the time of update. Cassandra provides the collection type of columns such as list, set and map.
- Some of its features are :
  a) Columns consist of various types, such as integer, big integer, text, float, double and Boolean.
  b) Cassandra also provides collection types such as set, list and map.
  c) Further, column values have an associated time stamp representing the time of update.
  d) This timestamp can be retrieved using the function write time.

2.7.3 Cassandra Clients

- Thrift is the driver-level interface; it provides the API for client implementations in a wide variety of languages. Thrift was developed at Facebook.
- A Client holds connections to a Cassandra cluster, allowing it to be queried. Each Client instance maintains multiple connections to the cluster nodes, provides policies to choose which node to use for each query and handles retries for failed query etc...
- Client instances are designed to be long-lived and usually a single instance is enough per application. As a given Client can only be "logged" into one keyspace at a time, it can make sense to create one client per keyspace used. This is however not necessary to query multiple keyspaces since it is always possible to use a single session with fully qualified table name in queries.
- The Cassandra cluster is denoted as a ring. The idea behind this representation is to show token distribution.

1. Write in action :

- To write, clients need to connect to any of the Cassandra nodes and send a write request. This node is called the **coordinator node**. When a node in Cassandra cluster receives a write request, it delegates it to a service called StorageProxy.
- This node may or may not be the right place to write the data to. The task of StorageProxy is to get the nodes (all the replicas) that are responsible to hold the data that is going to be written. It utilizes a replication strategy to do that.
- Once the replica nodes are identified, it sends the RowMutation message to them, the node waits for replies from these nodes, but it does not wait for all the replies to come.
- It only waits for as many responses as are enough to satisfy the client's minimum number of successful writes defined by consistency level.
- **Write operations at a node level :**
  - Each node processes the request individually. Every node first writes the mutation to the commit log and then writes the mutation to the memtable. Writing to the commit log ensures durability of the write as the memtable is an in-memory structure and is only written to disk when the memtable is flushed to disk.
  - A memtable is flushed to disk when :
    1. It reaches its maximum allocated size in memory
    2. The number of minutes a memtable can stay in memory elapses.
    3. Manually flushed by a user.
  - A memtable is flushed to an immutable structure called as **SSTable (Sorted String Table)**. The commit log is used for playback purposes in case data from the memtable is lost due to node failure.

2.8 Two Marks Questions with Answers

Q.1 What is consistency in a distributed system ?

Ans. : In a distributed system, consistency will be defined as one that responds with the same output for the same request at the same time across all the replicas.

Q.2 What is database Sharding ?

Ans. : Sharding is a method for distributing a single dataset across multiple databases, which can then be stored on multiple machines. This allows for larger datasets to be split into smaller chunks and stored in multiple data nodes, increasing the total storage capacity of the system.

Q.3 Why are NoSQL databases known as schemaless databases ?

Ans. : Because NoSQL databases are designed to store and query unstructured data, they do not require the same rigid schemas used by relational databases. Although a schema can be applied at the application level, NoSQL databases retain all of your unstructured data in its original raw format. This means that complete granularity is retained, even if you later change your application schema - Something that is simply not possible with a traditional SQL database.

Q.4 What is the difference between Sharding and replication ?

Ans. : Sharded database servers each contain a part of the overall data, i.e. they store different data on separate nodes. Replicated servers contain identical copies of the entire database.

Q.5 How is Sharding different from partitioning ?

Ans. : All partitions of a table reside on the same server whereas Sharding involves multiple servers. Therefore, Sharding implies a distributed architecture whereas partitioning does not. Partitions can be horizontal (split by rows) or vertical (by columns). Shards are usually only horizontal. In other words, all shards share the same schema but contain different records of the original table.

Q.6 What are write-write and read-write conflicts ?

Ans. : Write-write conflicts occur when two clients try to write the same data at the same time. Read-write conflicts occur when one client reads inconsistent data in the middle of another client's write.

Q.7 Define Cassandra.

Ans. : Cassandra is a distributed, fault tolerant, scalable, column oriented data store. Cassandra is a peer-to-peer distributed system made up of a cluster of nodes in which any node can accept a read or write request.

Q.8 What is the use of Bloom filters in Cassandra ?

Ans. : Bloom filters are used as a performance booster. Bloom filters are very fast, nondeterministic algorithms for testing whether an element is a member of a set. They are nondeterministic because it is possible to get a false-positive read from a Bloom filter, but not a false-negative. Bloom filters work by mapping the values in a data set into a bit array and condensing a larger data set into a digest string. Bloom filter is a special kind of cache.

Q.9 Explain sorted strings table.

Ans. : Sorted strings table which is a file format used by Cassandra to store the statics and the data from the memtables. The Cassandra SSTables are immutable hence any update on the table creates a new SSTables file. The data structure format used by SSTables is Log-Structured Merge which is qualified for writing intense heavy data sets compared to the traditional B tree structure.

Q.10 Explain Cassandra data center.

Ans. : Cassandra data center is the collection of related nodes which are configured in the cluster to perform the replication. The data centers can be physical data centers or logical data center and depending upon the workload a separate data center can be used.

Q.11 Explain advantages and disadvantages of graph data.

Ans. : • **Advantages of graph data :**

a) More descriptive queries

b) Greater flexibility in adapting your model

c) Greater performance when traversing data relationships.

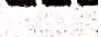
- Disadvantages of graph data stores :
  a) Difficult to scale,
  b) No standard language.

Q.12 Describe session consistency.

Ans. : Session consistency means read-your-writes consistency but at session level. Session can be identified with a conversation between a client and a server. As long as the conversation continues, we will read everything we have written during this conversation. If the session ends and we start another session with the same server, there is no guarantee that we can read values we have written during previous conversation.

Q.13 What are schemaless databases ?

Ans. : Schemaless databases are a type of NoSQL databases that do not have a predefined schema or structure for data. This means that data can be inserted and retrieved without adhering to a specific structure and the database can adapt to changes in data over time without requiring schema migrations or changes.



UNIT III

3 Basics of Hadoop

Syllabus

MapReduce workflows - unit tests with MRUnit - test data and local tests - anatomy of MapReduce job run - classic Map-reduce - YARN - failures in classic Map-reduce and YARN - job scheduling - shuffle and sort - task execution - MapReduce types - input formats - output formats.

Contents

3.1 Data Format

- The data is stored using a line-oriented ASCII format, in which each line is a record. The format supports a rich set of meteorological elements, many of which are optional or with variable data lengths.
- The default output format provided by hadoop is TextOutputFormat and it writes records as lines of text. If file output format is not specified explicitly, then text files are created as output files. Output key-value pairs can be of any format because TextOutputFormat converts these into strings with toString() method.
- HDFS data is stored in something called **blocks**. These blocks are the smallest unit of data that the file system can store. Files are processed and broken down into these blocks, which are then taken and distributed across the cluster and also replicated for safety.

3.1.1 Analyzing the Data with Hadoop

- Hadoop supports parallel processing, so we take this advantage for expressing query as a MapReduce job. After some local, small-scale testing, we will be able to run it on a cluster of machines.
- MapReduce works by breaking the processing into two phases : The map phase and the reduce phase.
- Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer. The programmer also specifies two functions : The map function and the reduce function.
- MapReduce is a method for distributing a task across multiple nodes. Each node processes the data stored on that node to the extent possible. A running MapReduce job consists of various phases which is described in the following Fig. 3.1.1.

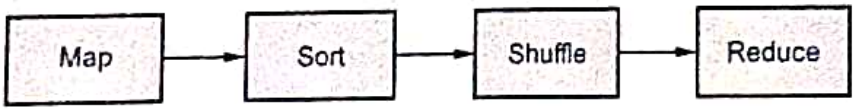


Fig. 3.1.1 Phases of Hadoop MapReduce

- In the map job, we split the input dataset into chunks. Map task processes these chunks in parallel. The map we use outputs as inputs for the reduced tasks. Reducers process the intermediate data from the maps into smaller tuples that reduce the tasks leading to the final output of the framework.
- The advantages of using MapReduce, which run over a distributed infrastructure like CPU and storage are automatic parallelization and distribution of data in

blocks across a distributed system, fault-tolerance against failure of storage, computation and network infrastructure, deployment, monitoring and security capability and clear abstraction for programmers. Most MapReduce programs are written in Java. It can also be written in any scripting language using the Streaming API of Hadoop.

3.1.2 Scaling Out

- To scale out, we need to store the data in a distributed file system, typically HDFS, to allow Hadoop to move the MapReduce computation to each machine hosting a part of the data.
- A MapReduce job is a unit of work that the client wants to be performed : It consists of the input data, the MapReduce program and configuration information.
- Hadoop runs the job by dividing it into tasks, of which there are two types : Map tasks and reduce tasks.
- There are two types of nodes that control the job execution process : A job tracker and a number of task trackers.
- **Job tracker :** This tracker plays the role of scheduling jobs and tracking all jobs assigned to the task tracker.
- **Task tracker :** This tracker plays the role of tracking tasks and reporting the status of tasks to the job tracker.
- Hadoop divides the input to a MapReduce job into fixed-size pieces called **input splits**. Hadoop creates one map task for each split, which runs the user defined map function for each record in the split.
- That split information is used by YARN ApplicationMaster to try to schedule map tasks on the same node where split data is residing thus making the task data local. If map tasks are spawned at random locations then each map task has to copy the data it needs to process from the DataNode where that split data is residing, resulting in lots of cluster bandwidth. By trying to schedule map tasks on the same node where split data is residing, what Hadoop framework does is to send computation to data rather than bringing data to computation, saving cluster bandwidth. This is called data locality optimization.
- Map tasks write their output to the local disk, not to HDFS. Why is this? Map output is intermediate output : It is processed by reducing tasks to produce the final output and once the job is complete the map output can be thrown away. So storing it in HDFS, with replication, would be overkill. If the node running the map task fails before the map output has been consumed by the reduce task, then

Hadoop will automatically rerun the map task on another node to re-create the map output.

- Fig. 3.1.2 shows MapReduce data flow with a single reduce task.

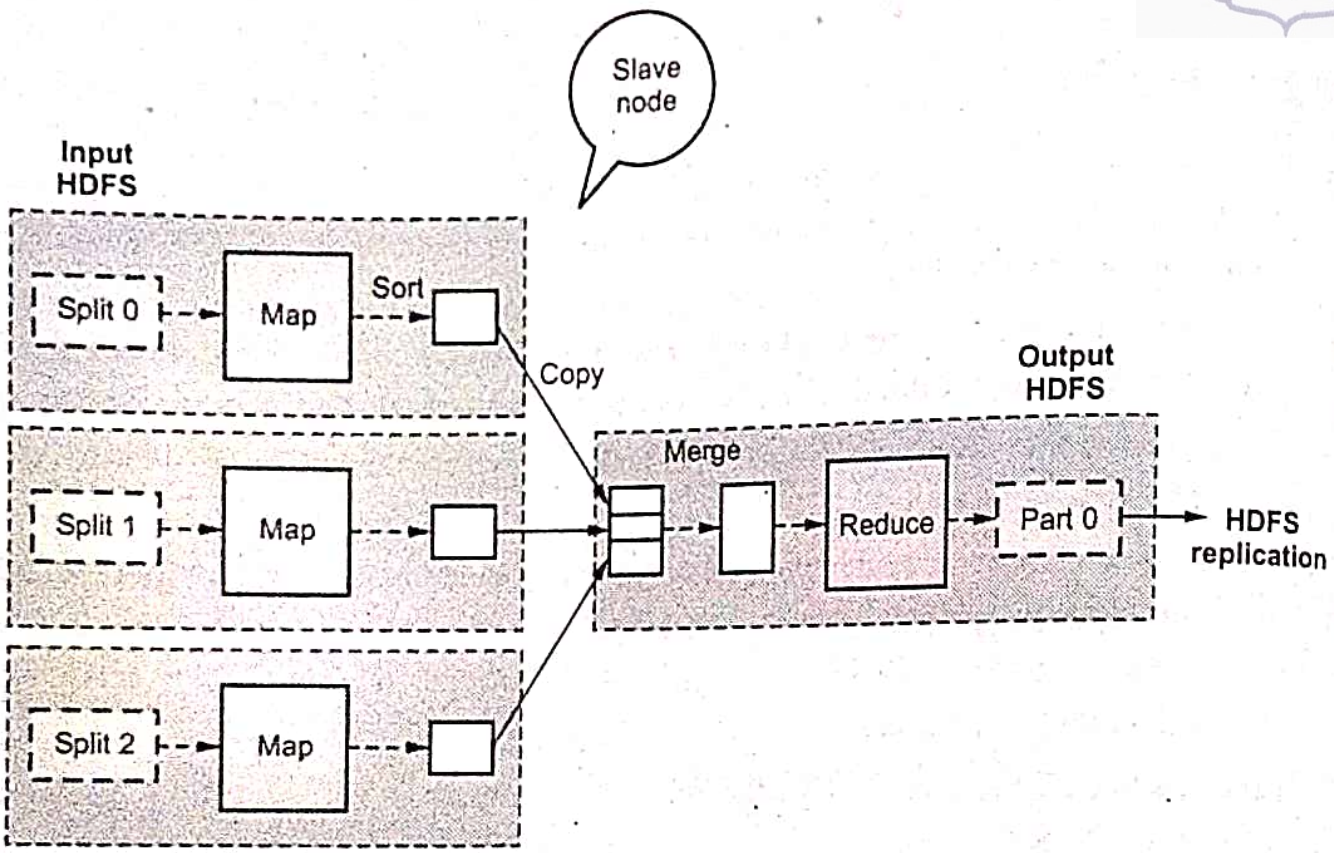


Fig. 3.1.2 MapReduce data flow with a single reduce task

- The number of reduced tasks is not governed by the size of the input, but is specified independently.
- When there are multiple reducers, the map tasks partition their output, each creating one partition for each reduce task. There can be many keys in each partition, but the records for any given key are all in a single partition.
- Fig. 3.1.3 shows MapReduce data flow with multiple reduce tasks.
- Hadoop allows the user to specify a combiner function to be run on the map output, the combiner function's output forms the input to the reduce function. Since the combiner function is an optimization, Hadoop does not provide a guarantee of how many times it will call it for a particular map output record, if at all.

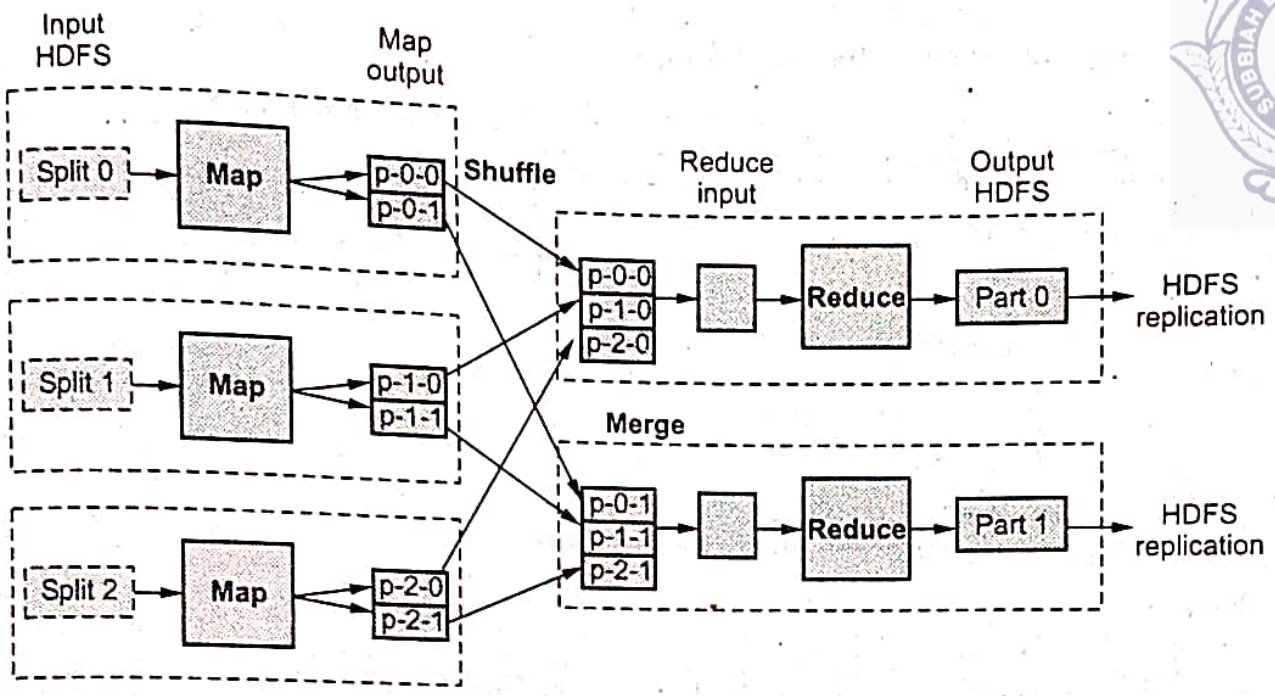


Fig. 3.1.3 MapReduce data flow with multiple reduce tasks

3.2 Hadoop Streaming

- Hadoop Streaming is an API that allows writing Mappers and Reduces in any language. It uses UNIX standard streams as the interface between Hadoop and the user application. Hadoop streaming is a utility that comes with the Hadoop distribution.
- Streaming is naturally suited for text processing. The data view is line-oriented and processed as a key-value pair separated by a 'tab' character. The Reduce function reads lines from the standard input, which is sorted by key, and writes its results to the standard output.
- It helps in real-time data analysis, which is much faster using MapReduce programming running on a multi-node cluster. There are different Technologies like spark Kafka and others which help in real-time Hadoop streaming.
- Features of Hadoop Streaming
  1. Users can execute non-Java-programmed MapReduce jobs on Hadoop clusters. Supported languages include Python, Perl, and C++.
  2. Hadoop Streaming monitors the progress of jobs and provides logs of a job's entire execution for analysis.
  3. Hadoop Streaming works on the MapReduce paradigm, so it supports scalability, flexibility, and security/authentication.
  4. Hadoop Streaming jobs are quick to develop and don't require much programming.

- Following code shows Streaming Utility :

```
> hadoop jar <dir>/hadoop-*streaming*.jar \
    -file /path/to/mapper.py \
    -mapper /path/to/mapper.py \
    -file /path/to/reducer.py \
    -reducer /path/to/reducer.py \
    -input /user/hduser/books/* \
    -output /user/hduser/books-output
```

Path to the streamingjar library

Location of mapperfile and define it as mapper

Location of reducerfile and define it as reducer

Input and output locations

Where :

Input = Input location for Mapper from where it can read input

Output = Output location for the Reducer to store the output

Mapper = The executable file of the Mapper

Reducer = The executable file of the Reducer

- Fig. 3.2.1 shows code execution process.

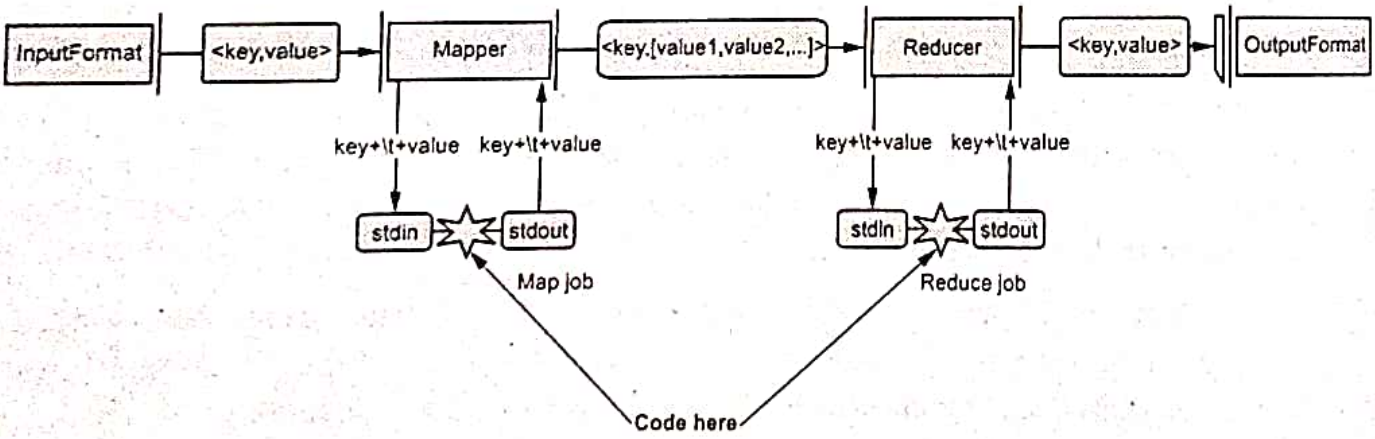


Fig. 3.2.1 Code execution process

- Map and reduce functions read their input from STDIN and produce their output to STDOUT. In the diagram above, the Mapper reads the input data from Input Reader/Format in the form of key-value pair, maps them as per logic written on code, and then passes through the Reduce stream, which performs data aggregation and releases the data to the output.

3.3 Hadoop Pipes

- Hadoop pipes is the name of the C++ interface to Hadoop MapReduce. Unlike Streaming, this uses standard input and output to communicate with the map and reduce code.
- Pipes uses sockets as the channel over which the task tracker communicates with the process running the C++ map or reduce function.
- Fig. 3.3.1 shows execution of streaming and pipes.

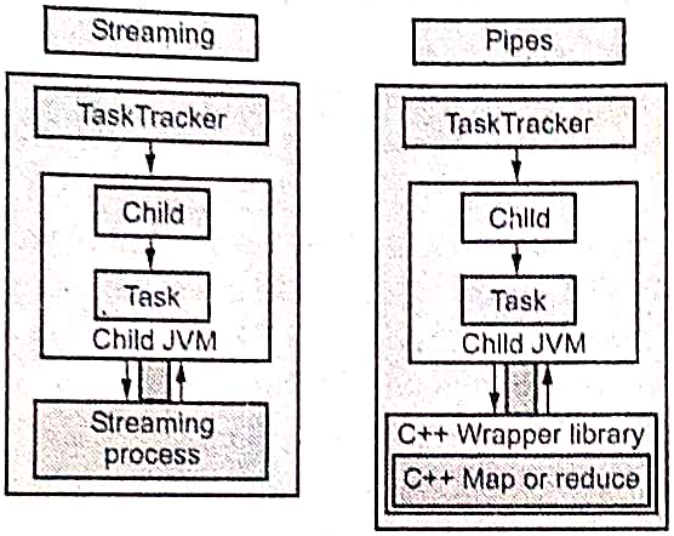


Fig. 3.3.1 Execution of streaming and pipes

- With Hadoop pipes, we can implement applications that require higher performance in numerical calculations using C++ in MapReduce. The pipes utility works by establishing a persistent socket connection on a port with the Java pipes task on one end, and the external C++ process at the other.
- Other dedicated alternatives and implementations are also available, such as Pydoop for Python, and libhdfs for C. These are mostly built as wrappers, and are JNI-based. It is, however, noticeable that MapReduce tasks are often a smaller component to a larger aspect of chaining, redirecting and recursing MapReduce jobs. This is usually done with the help of higher-level languages or APIs like Pig, Hive and Cascading, which can be used to express such data extraction and transformation problems.

3.4 Design of Hadoop Distributed File System (HDFS)

- The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. HDFS is the file system component of Hadoop.
- HDFS stores file system metadata and application data separately. As in other distributed file systems, like GFS , HDFS stores metadata on a dedicated server, called the NameNode. Application data is stored on other servers called

DataNodes. All servers are fully connected and communicate with each other using TCP-based protocols.

- Hadoop Distributed File System (HDFS) is a distributed file system that handles large data sets running on commodity hardware. It is used to scale a single Apache Hadoop cluster to hundreds of nodes.
- A block is the minimum amount of data that it can read or write. HDFS blocks are 128 MB by default and this is configurable. When a file is saved in HDFS, the file is broken into smaller **chunks** or "**blocks**".
- HDFS is a fault-tolerant and resilient system, meaning it prevents a failure in a node from affecting the overall system's health and allows for recovery from failure too. In order to achieve this, data stored in HDFS is automatically replicated across different nodes.
- HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file.
- Hadoop distributed file system is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines.
- Apache Hadoop HDFS architecture follows a master/slave architecture, where a cluster comprises of a single NameNode (MasterNode) and all the other nodes are DataNodes (Slave nodes).
- HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.
- Design issue of HDFS :
  1. Commodity hardware : HDFS do not require expensive hardware for executing user tasks. It's designed to run on clusters of commodity hardware.
  2. Streaming data access : HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern.
  3. Multiple writers, arbitrary file modifications : Files in HDFS may be written to by a single writer. Writes are always made at the end of the file. There is no support for multiple writers, or for modifications at arbitrary offsets in the file.
  4. Low-latency data access.
  5. Holds lots of small files.
  6. Store very large files.

- The HDFS achieve the following goals :
  1. **Manage large datasets :** Organizing and storing datasets can be a hard talk to handle. HDFS is used to manage the applications that have to deal with huge datasets. To do this, HDFS should have hundreds of nodes per cluster.
  2. **Detecting faults :** HDFS should have technology in place to scan and detect faults quickly and effectively as it includes a large number of commodity hardware. Failure of components is a common issue.
  3. **Hardware efficiency :** When large datasets are involved it can reduce the network traffic and increase the processing speed.

3.4.1 HDFS Architecture

- Fig. 3.4.1 shows HDFS architecture.

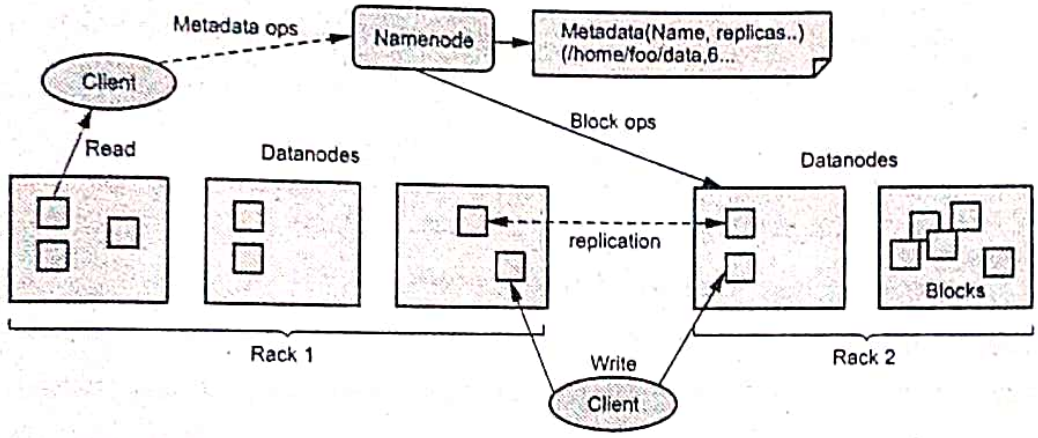


Fig. 3.4.1 Hadoop architecture

- Hadoop distributed file system is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines.
- Apache Hadoop HDFS architecture follows a master/slave architecture, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes).
- DataNodes process and store data blocks, while NameNodes manage the many DataNodes, maintain data block metadata and control client access.

1. NameNode and DataNode

- Namenode holds the meta data for the HDFS like Namespace information, block information etc. When in use, all this information is stored in main memory. But this information also stored in disk for persistence storage.

- Namenode manages the file system namespace. It keeps the directory tree of all files in the file system and metadata about files and directories.
- DataNode is a slave node in HDFS that stores the actual data as instructed by the NameNode. In brief, NameNode controls and manages a single or multiple data nodes.
- DataNode serves to read or write requests. It also creates, deletes and replicates blocks on the instructions from the NameNode.
- Fig. 3.4.2 shows Namenode. It shows how NameNode stores information on disk.

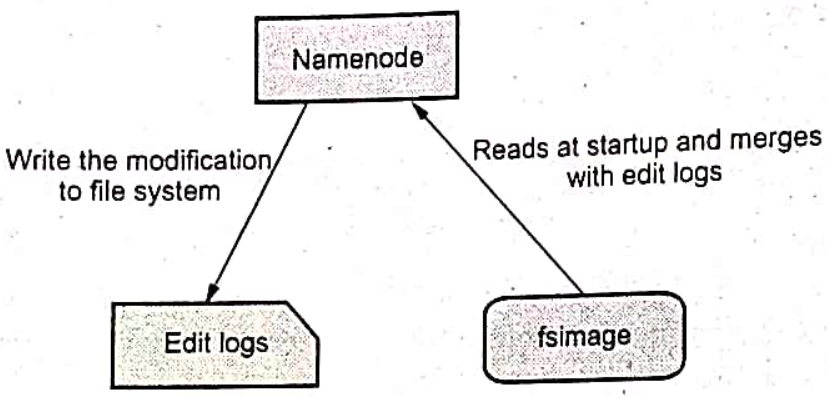


Fig. 3.4.2 Name node

- Two different files are :
  1. **fsimage :** It is the snapshot of the file system when name node started.
  2. **Edit logs :** It is the sequence of changes made to the file system after name node started.
- Only in the restart of namenode, edit logs are applied to fsimage to get the latest snapshot of the file system.
- But namenode restart are rare in production clusters which means edit logs can grow very large for the clusters where namenode runs for a long period of time.
- The following issues we will encounter in this situation :
  1. Editlog become very large, which will be challenging to manage it.
  2. Namenode restart takes long time because lot of changes to be merged.
  3. In the case of crash, we will lost huge amount of metadata since fsimage is very old.
- So to overcome this issues we need a mechanism which will help us reduce the edit log size which is manageable and have up to date fsimage, so that load on namenode reduces.

- Secondary Namenode helps to overcome the above issues by taking over responsibility of merging editlogs with fsimage from the namecode.
- Fig. 3.4.3 shows secondary Namenode.

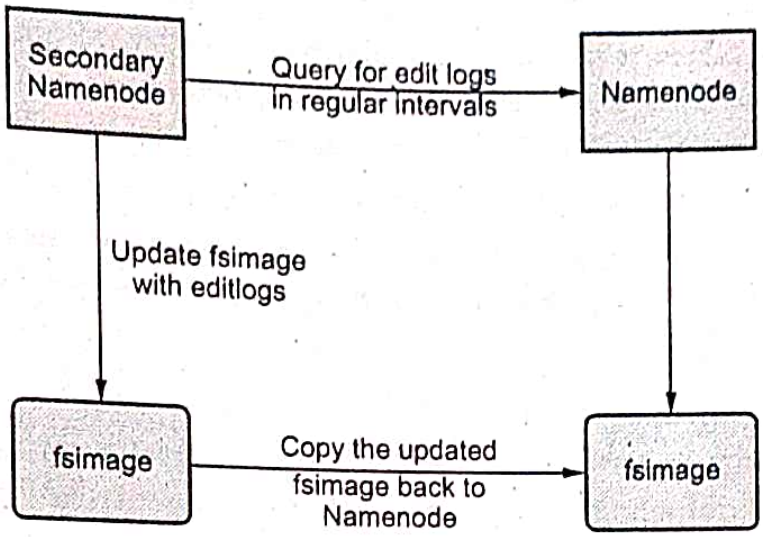


Fig. 3.4.3 Secondary Namenode

- Working of secondary Namenode :
  1. It gets the edit logs from the Namenode in regular intervals and applies of fsimage.
  2. Once it has new fsimage, it copies back to Namenode.
  3. Namenode will use this fsimage for the next restart, which will reduce the startup time.
- Secondary Namenode's whole purpose is to have a checkpoint in HDFS. Its just a helper node for Namecode. That is why it also known as **checkpoint node** inside the community.

3.4.2 HDFS Block

- HDFS is a block structured file system. In general the users data stored in HDFS in terms of block. The files in the file system are divided into one or more segments called **blocks**. The default size of HDFS block is 64 MB that can be increased as per need.
- The HDFS is fault tolerant such that if a data node fails then the current block write operation on the data node is re-replicated to some other node. The block size, number of replicas and replication factors are specified in the Hadoop configuration file. The synchronization between name node and data node is done by heartbeat functions which are periodically generated by data node to name node.

- Apart from above components the job tracker and task trackers are used when map reduce applications run over the HDFS. Hadoop Core consists of one master job tracker and several task trackers. The job tracker runs on name nodes like a master while task trackers run on data nodes like slaves.
- The job tracker is responsible for taking the requests from a client and assigning task trackers to it with tasks to be performed. The job tracker always tries to assign tasks to the task tracker on the data nodes where the data is locally present.
- If for some reason the node fails the job tracker assigns the task to another task tracker where the replica of the data exists since the data blocks are replicated across the data nodes. This ensures that the job does not fail even if a node fails within the cluster.

The Command - Line Interface :

- The HDFS can be manipulated either using the command line. All the commands used for manipulating HDFS through the command line interface begin with the "hadoop fs" command.
- Most of the Linux commands are supported over HDFS which starts with "-"sign.
- For example : The command for listing the files in Hadoop directory will be,
  #hadoop fs -ls
- The general syntax of HDFS command line manipulation is,
  #hadoop fs -<command>

3.4.3 Java Interface

- Hadoop is written in Java, so most Hadoop filesystem interactions are mediated through the Java API. The filesystem shell, for example, is a Java application that uses the Java FileSystem class to provide file system operations. By exposing its filesystem interface as a Java API, Hadoop makes it awkward for non-Java applications to access.

1. Reading Data from a Hadoop URL :

- To read a file from a Hadoop file system is by using a **java.net.URL** object to open a stream to read the data. The syntax is as follows :

```
InputStream in = null;
                try {
                        in = new URL("hdfs://host/path").openStream();
                        // process in
                    }
                    finally {
                            IOUtils.closeStream(in);
                        }
```

- Java recognizes Hadoop's **hdfs** URL scheme by calling the **setURLStreamHandlerFactory** method on URL with an instance of **FsUrlStreamHandlerFactory**.
- The `setURLStreamHandlerFactory` is a method in the `java.net.URL` class that sets the URL stream handler factory for the Java Virtual Machine. This factory is responsible for creating URL stream handler instances that are used to retrieve the contents of a URL.
- This method can only be called once per JVM, so it is typically executed in a static block.
- **Example : Displaying files from a Hadoop file system on standard output using a URLStreamHandler.**

```
import java.io.InputStream;
import java.net.URL;
import org.apache.hadoop.fs.FsUrlStreamHandlerFactory;
import org.apache.hadoop.io.IOUtils;
// vv URLCat
public class URLCat {
 static {
  URL.setURLStreamHandlerFactory(new FsUrlStreamHandlerFactory());
 }
  public static void main(String[] args) throws Exception {
  InputStream in = null;
  try {
   in = new URL(args[0]).openStream();
   IOUtils.copyBytes(in, System.out, 4096, false);
  } finally {
   IOUtils.closeStream(in);
  }
 }
}
```

2. Reading Data Using the FileSystem API :

- Sometimes it is not possible to set **URLStreamHandlerFactory** for our application, so in that case we will use Filesystem API for opening an input stream for a file. A file in a Hadoop filesystem is represented by a Hadoop Path object.
- There are two static factory methods for getting a FileSystem instance :

```
public static FileSystem get(Configuration conf) throws IOException
public static FileSystem get(URI uri, Configuration conf) throws IOException
```

- FileSystem is a generic abstract class used to interface with a file system. FileSystem class also serves as a factory for concrete implementations, with the following methods :

  Public static FileSystem get(Configuration conf) : Use information from configuration such as scheme and authority.

- A configuration object encapsulates a client or server's configuration, which is set using configuration files read from the classpath, such as **conf/core-site.xml**.

FSDataInputStream :

- The open () method on FileSystem actually returns a FSDataInputStream rather than a standard java.io class.
- This class is a specialization of java.io.DataInputStream with support for random access, so we can read from any part of the stream : Package org.apache.hadoop.fs;

Writing Data :

- The FileSystem class has a number of methods for creating a file. The simplest is the method that takes a path object for the file to be created and returns an output stream to write to : public FSDataOutputStream create(Path f) throws IOException FSDataOutputStream
- The create() method on FileSystem returns an FSDataOutputStream, which, like FSDataInputStream, has a method for querying the current position in the file : package org.apache.hadoop.fs;
- We can append to an existing file using the append() method :

public FSDataOutputStream append(Path f) throws IOException

- The append operation allows a single writer to modify an already written file by opening it and writing data from the final offset in the file. With this API applications that produce unbounded files, such as log files, can write to an existing file after a restart, for example, the append operation is optional and not implemented by all Hadoop filesystems.

3.4.4 Data Flow

1. Anatomy of a File Read:

- Fig. 3.4.4 shows sequence of events when reading a file. It shows data flows between client and HDFS, the namenode and the datanodes.
- The client opens the file it wishes to read by calling open() on the FileSystem object, which for HDFS is an instance of Distributed FileSystem (DFS). DFS calls the namenode, using RPC, to determine the locations of the blocks for the first few blocks in the file.

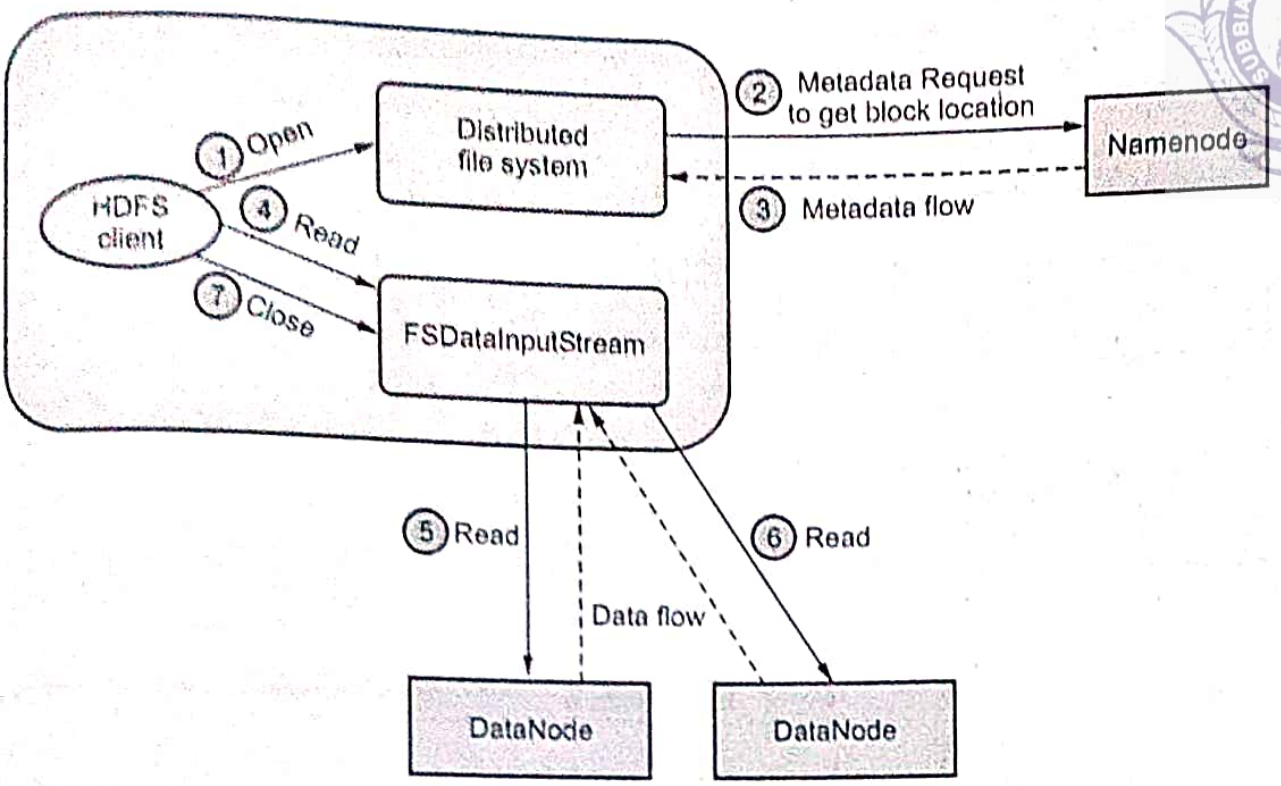


Fig. 3.4.4 Client reading data from HDFS

- For each block, the namenode returns the addresses of the datanodes that have a copy of that block. Furthermore, the datanodes are sorted according to their proximity to the client. If the client is itself a datanode, then it will read from the local datanode, if it hosts a copy of the block.
- The DFS returns an FSDataInputStream to the client for it to read data from. FSDataInputStream in turn wraps a DFSInputStream, which manages the datanode and namenode I/O.
- The client then calls read() on the stream. DFSInputStream, which has stored the datanode addresses for the first few blocks in the file, then connects to the first (closest) datanode for the first block in the file.
- Data is streamed from the datanode back to the client, which calls read() repeatedly on the stream. When the end of the block is reached, DFSInputStream will close the connection to the datanode, then find the best datanode for the next block. This happens transparently to the client, which from its point of view is just reading a continuous stream.
- Blocks are read in order with the DFSInputStream opening new connections to datanodes as the client reads through the stream. It will also call the namenode to retrieve the datanode locations for the next batch of blocks as needed. When the client has finished reading, it calls close() on the FSDataInputStream.
- During reading, if the DFSInputStream encounters an error while communicating with a datanode, then it will try the next closest one for that block.

2. Anatomy of a File Write:

- Fig. 3.4.5 shows anatomy of a file write.

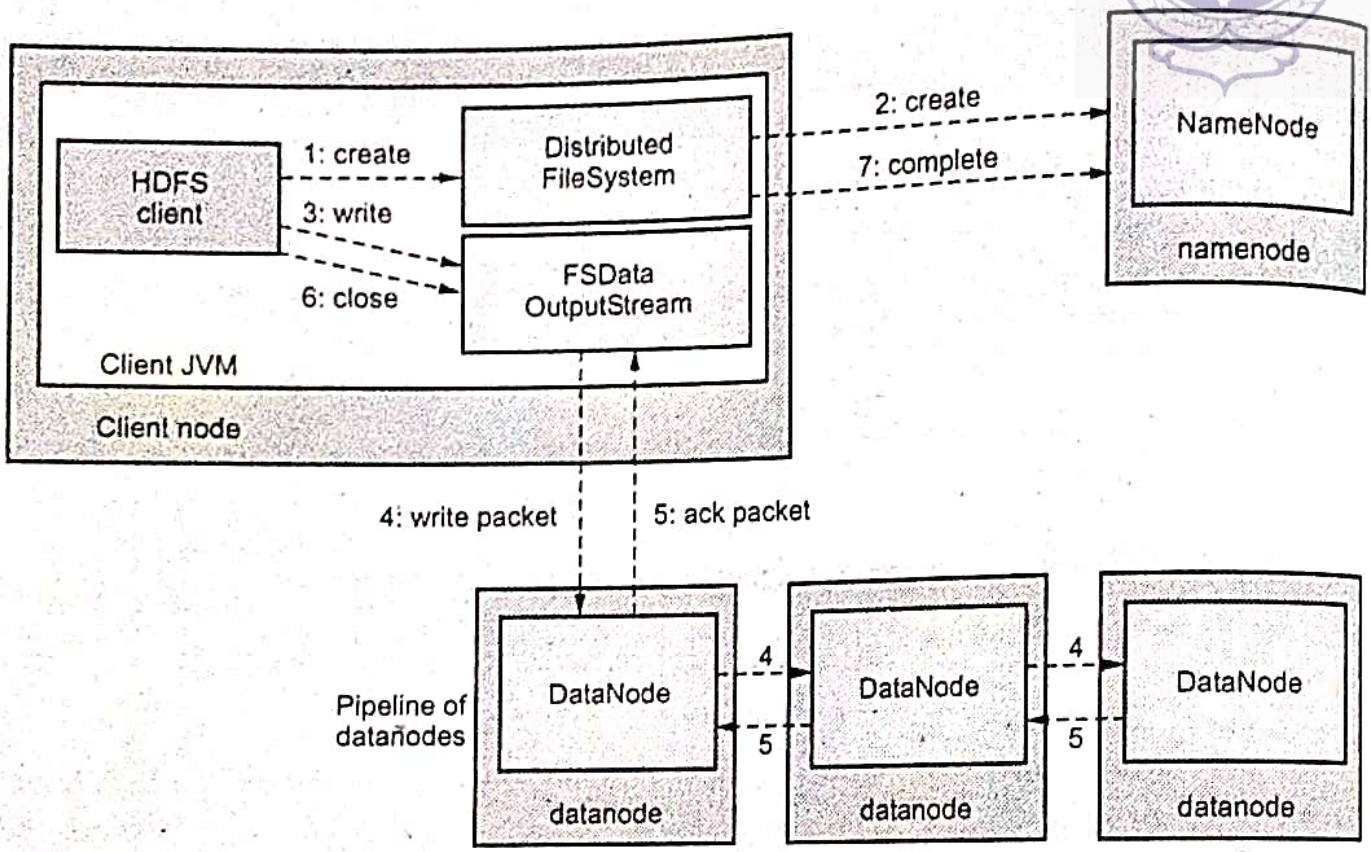


Fig. 3.4.5 Anatomy of a file write

1. The client calls create() on DistributedFileSystem to create a file.
2. An RPC call to the namenode happens through the DFS to create a new file.
3. As the client writes data, data is split into packets by DFSOutputStream, which is then written to an internal queue, called **data queue**. Datastreamer consumes the data queue.
4. Data streamer streams the packets to the first DataNode in the pipeline. It stores the packet and forwards it to the second DataNode in the pipeline.
5. In addition to the internal queue, DFSOutputStream also manages the "Ackqueue" of the packets that are waiting to be acknowledged by DataNodes.
6. When the client finishes writing the file, it calls close() on the stream.

3.4.5 Heartbeat Mechanism in HDFS

- Heartbeat is a single indicating that is is alive. A datanode sends heartbeat to Namenode and task tracker will send its heartbeat to job tracker.
- Fig. 3.4.6 shows heartbeat mechanism.

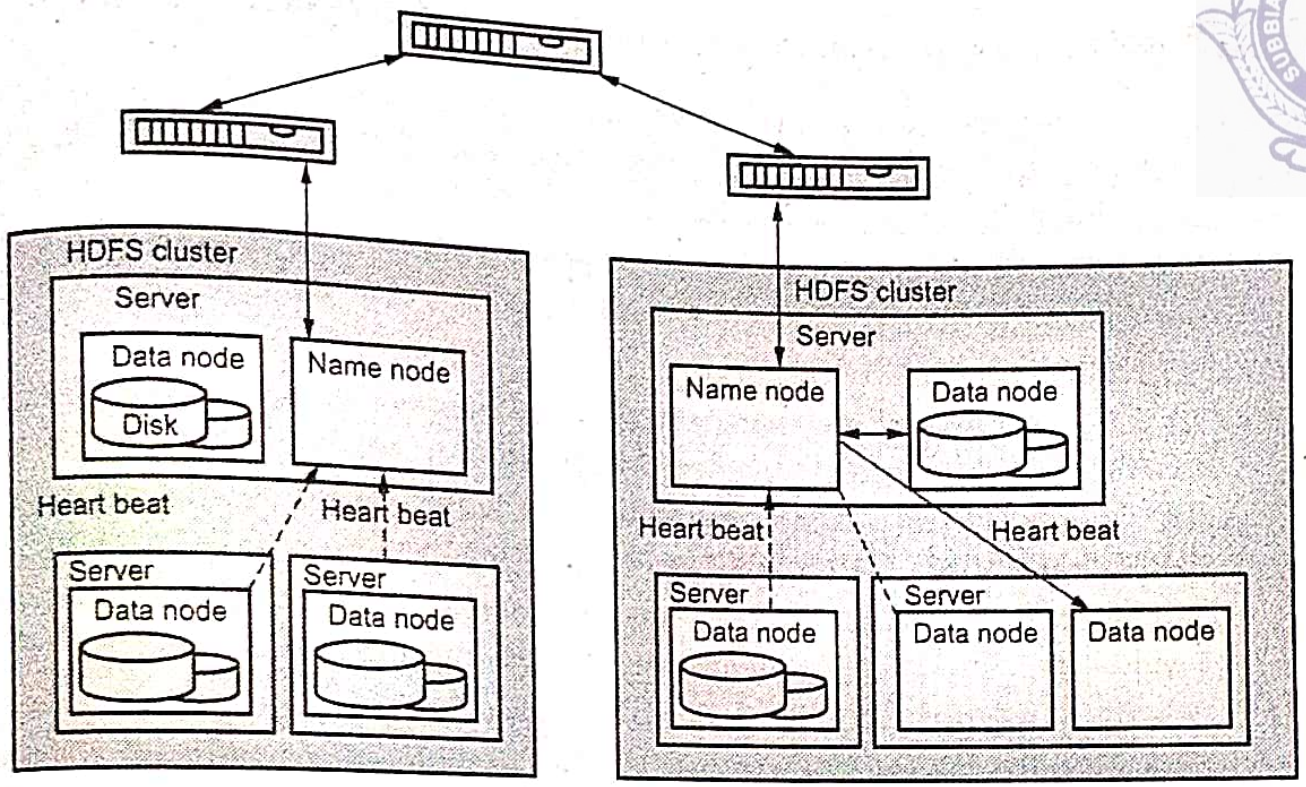


Fig. 3.4.6 Heartbeat mechanism

- The connectivity between the NameNode and a DataNode are managed by the persistent heartbeats that are sent by the DataNode every three seconds.
- The heartbeat provides the NameNode confirmation about the availability of the blocks and the replicas of the DataNode.
- Additionally, heartbeats also carry information about total storage capacity, storage in use and the number of data transfers currently in progress. These statistics are by the NameNode for managing space allocation and load balancing.
- During normal operations, if the NameNode does not receive a heartbeat from a DataNode in ten minutes the NameNode, it considers that DataNode to be out of service and the block replicas hosted to be unavailable.
- The NameNode schedules the creation of new replicas of those blocks on other DataNodes.
- The heartbeats carry roundtrip communications and instructions from the NameNode, including commands to :
  a) Replicate blocks to other nodes.
  b) Remove local block replicas.
  c) Re-register the node.
  d) Shut down the node.
  e) Send an immediate block report.

3.4.6 Role of Sorter, Shuffler and Combiner in MapReduces Paradigm

- A combiner, also known as a **semi-reducer**, is an optional class that operates by accepting the inputs from the Map class and thereafter passing the output key-value pairs to the Reducer class.
- The main function of a combiner is to summarize the map output records with the same key. The output of the combiner will be sent over the network to the actual reducer task as input.
- The process of transferring data from the mappers to reducers is known as **shuffling** i.e. the process by which the system performs the sort and transfers the map output to the reducer as input. So, shuffle phase is necessary for the reducers, otherwise, they would not have any input.
- Shuffle phase in Hadoop transfers the map output from Mapper to a Reducer in MapReduce. Sort phase in MapReduce covers the merging and sorting of map outputs.
- Data from the mapper are grouped by the key, split among reducers and sorted by the key. Every reducer obtains all values associated with the same key, Shuffle and sort phase in Hadoop occur simultaneously and are done by the MapReduce framework.

3.5 Hadoop I/O

- Hadoop input output system comes with a set of primitives. Hadoop deals with multi-terabytes of datasets; a special consideration on these primitives will give an idea how Hadoop handles data input and output.

3.5.1 Data Integrity

- Data integrity means that data should remain accurate and consistent all across its storing, processing and retrieval operations.
- However, since every I/O operation on the disk or network carries with it a small chance of introducing errors into the data that it is reading or writing. The usual way of detecting corrupted data is by computing a checksum for the data when it first enters the system and again whenever it is transmitted across a channel that is unreliable and hence capable of corrupting the data.
- The commonly used error detecting code is CRC-32 which computes a 32-bit integer checksum input of any size.

Data Integrity in HDFS :

- HDFS transparently checksums all data written to it and by default verifies checksums when reading data. A separate checksum is created for every

io.bytes.per.checksum bytes of data. The default is 512 bytes, and since a CRC-32 checksum is 4 bytes long, the storage overhead is not an issue.

- All data that enters into the system is verified by the datanodes before being forwarded for storage or further processing. Data sent to the datanode pipeline is verified through checksums and any corruption found is immediately notified to the client with **ChecksumException.**
- The client read from the datanode also goes through the same drill. The datanodes maintain a log of checksum verification to keep track of the verified block. The log is updated by the datanode upon receiving a block verification success signal from the client. This type of statistics helps in keeping the bad disks at bay.
- Apart from this, a periodic verification on the block store is made with the help of DataBlockScanner running along with the datanode thread in the background. This protects data from corruption in the physical storage media.
- HDFS stores replicas of blocks, it can "heal" corrupted blocks by copying one of the good replicas to produce a new, uncorrupt replica.
- If a client detects an error when reading a block, it reports the bad block and the datanode it was trying to read from to the namenode before throwing a ChecksumException. The namenode marks the block replica as corrupt, so it does not direct clients to it, or try to copy this replica to another datanode.
- It then schedules a copy of the block to be replicated on another datanode, so its replication factor is back at the expected level. Once this has happened, the corrupt replica is deleted. It is possible to disable verification of checksums by passing false to the setVerify Checksum() method on FileSystem, before using the open() method to read a file.

3.5.2 Hadoop Local File System

- The Hadoop local file system performs client side checksums. When a file is created, it automatically creates a transparent file in the background with the file **name.crc**, which uses check chunks to check the file.
- Each chunk can check a segment up to 512 bytes, the chunk of data is divided by the **file.byte-per-checksum** property and the chunk is then stored as metadata in a .crc file.
- The file can be read correctly though the settings of the files might change and if an error is detected then the local system throws a checksum exception.

Checksum file system :

- Local file systems use checksum file systems as a security measure to ensure that the data is not corrupt or damaged in any way.

- In this file system, the underlying file system is called the **raw file system**, if an error is detected while working on the checksum file system, it will call **reportchecksumfailure()**.
- Here the local system moves the affected file to another directory as a file titled as bad_file. It is then the responsibility of an administrator to keep a check on these bad_files and take the necessary action.

3.5.3 Compression

- Compression has two major benefits :
  a) It creates space for a file.
  b) It also increases the speed of data transfer to a disk or drive.
- The following are the commonly used methods of compression in Hadoop : a) Deflate, b) Gzip, c) Bzip2, d) Lzo, e) Lz4, and f) Snappy.
- All these compression methods primarily provide optimization of speed and storage space and they all have different characteristics and advantages.
- Gzip is a general compressor used to clear the space and performs faster than bzip2, but the decompression speed of bzip2 is good.
- Lzo, Lz4 and Snappy can be optimized as required and hence, are the better tools in comparison to the others.

Codecs :

- A codec is an algorithm that is used to perform compression and decompression of a data stream to transmit or store it.
- In Hadoop, these compression and decompression operations run with different codecs and with different compression formats.

3.5.4 Serialization

- Serialization is the process of converting a data object, a combination of code and data represented within a region of data storage into a series of bytes that saves the state of the object in an easily transmittable form. In this serialized form, the data can be delivered to another data store, application, or some other destination.
- Data serialization is the process of converting an object into a stream of bytes to more easily save or transmit it.
- Fig. 3.5.1 shows serialization and deserialization.
- The reverse process, constructing a data structure or object from a series of bytes is deserialization. The deserialization process recreates the object, thus making the data easier to read and modify as a native structure in a programming language.

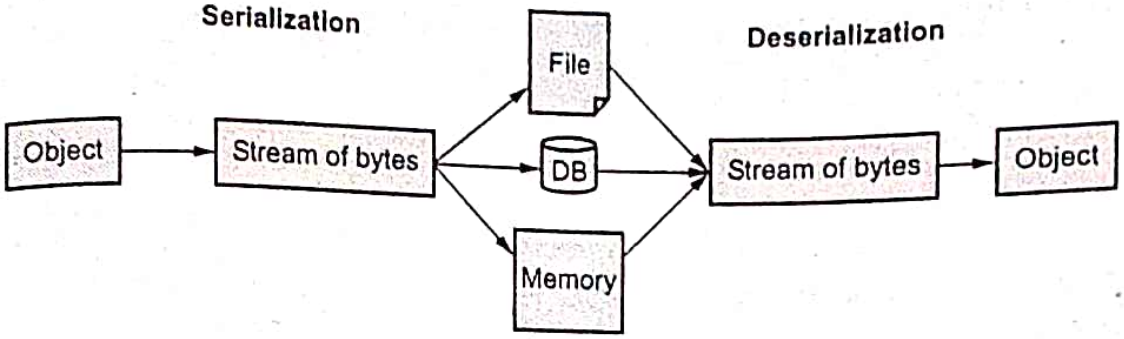


Fig. 3.5.1 Serialization and deserialization

- Serialization and deserialization work together to transform/recreate data objects to/from a portable format.
- Serialization enables us to save the state of an object and recreate the object in a new location. Serialization encompasses both the storage of the object and exchange of data. Since objects are composed of several components, saving or delivering all the parts typically requires significant coding effort, so serialization is a standard way to capture the object into a shareable format.
- Serialization is divided in two methods of data processing : Intercrossing communication and data storage.
- Intercrossing communication between nodes is processing that uses remote procedure calls (RPC's). In RPC, the data is converted to the binary system and is then transferred to a remote node where the data is de-serialized into the original message. The lifespan of RPC is less than a second.
- It is desirable that an RPC serialization format is :
  a) **Compact :** A compact format makes the best use of network bandwidth, which is the most scarce resource in a data center.
  b) **Fast :** Interprocess communication forms the backbone for a distributed system, so it is essential that there is as little performance overhead as possible for the serialization and deserialization process.
  c) **Extensible :** Protocols change over time to meet new requirements, so it should be straightforward to evolve the protocol in a controlled manner for clients and servers.
  d) **Interoperable :** For some systems, it is desirable to be able to support clients that are written in different languages to the server, so the format needs to be designed to make this possible.

3.5.5 The Writable Interface

- Hadoop uses its own serialization format called **Writable**. It is written in Java and is fast as well as compact. The other serialization framework supported by Hadoop is Avro.
- The Writable interface defines two methods : One for writing its state to a DataOutput binary stream and one for reading its state from a DataInput binary stream.
- When we write a key as IntWritable in the Mapper class and send it to the reducer class, there is an intermediate phase between the Mapper and Reducer class i.e., shuffle and sort, where each key has to be compared with many other keys. If the keys are not comparable, then the shuffle and sort phase won't be executed or may be executed with a high amount of overhead.
- If a key is taken as IntWritable by default, then it has a comparable feature because of RawComparator acting on that variable. It will compare the key taken with the other keys in the network. This cannot take place in the absence of Writable.
- **WritableComparator** is a general-purpose implementation of RawComparator for WritableComparable classes. It provides two main functions :
  a) It provides a default implementation of the raw compare() method that deserializes the objects to be compared from the stream and invokes the object compare() method.
  b) It acts as a factory for RawComparator instances.
- To provide mechanisms for serialization and deserialization of data, Hadoop provided two important interfaces Writable and WritableComparable. Writable interface specification is as follows :

```
package org.apache.hadoop.io;
 import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
 public interface Writable
{
 void write(DataOutput out) throws IOException;
 void readFields(DataInput in) throws IOException;
}
```

- WritableComparable interface is sub-interface of Hadoop's Writable and Java's Comparable interfaces. Its specification is shown below :

```
public interface WritableComparable extends Writable, Comparable
{
}
```

Writable Classes - Hadoop Data Types :

- Hadoop provides classes that wrap the Java primitive types and implement the WritableComparable and Writable interfaces. They are provided in the **org.apache.hadoop.io package.**
- All the Writable wrapper classes have a get() and a set() method for retrieving and storing the wrapped value.

Primitive Writable Classes :

- These are writable wrappers for Java primitive data types and they hold a single primitive value that can be set either at construction or via a setter method.
- All these primitive writable wrappers have get() and set() methods to read or write the wrapped value. Below is the list of primitive writable data types available in Hadoop.

a) BooleanWritable b) ByteWritable

c) IntWritable d) VIntWritable

e) FloatWritable f) LongWritable

g) VLongWritable h) DoubleWritable.

- In the above list VIntWritable and VLongWritable are used for variable length Integer types and variable length long types respectively.
- Serialized sizes of the above primitive writable data types are the same as the size of actual java data types. So, the size of IntWritable is 4 bytes and LongWritable is 8 bytes.

Text :

- Text is a Writable for UTF-8 sequences. It can be thought of as the Writable equivalent of **java.lang.String.**
- The Text class uses an int to store the number of bytes in the string encoding, so the maximum value is 2 GB.

BytesWritable :

- BytesWritable is a wrapper for an array of binary data. Its serialized format is an integer field (4 bytes) that specifies the number of bytes to follow, followed by the bytes themselves.
- BytesWritable is mutable and its value may be changed by calling its set() method.
- NullWritable is a special type of writable, as it has a zero-length serialization. No bytes are written to, or read from, the stream. It is used as a placeholder.

ObjectWritable and GenericWritable

ObjectWritable is a general-purpose wrapper for the following : Java primitives, string, enum, writable, null, or arrays of any of these types.

- It is used in Hadoop RPC to marshal and unmarshal method arguments and return types.
- There are four **writable** collection types in the org.apache.hadoop.io package: Array Writable, TwoDArrayWritable, MapWritable, and SortedMapWritable.
- ArrayWritable and TwoDArrayWritable are Writable implementations for arrays and two-dimensional arrays of Writable instances. All the elements of an ArrayWritable or a TwoDArrayWritable must be instances of the same class.
- ArrayWritable and TwoDArrayWritable both have get() and set() methods, as well as a toArray() method, which creates a shallow copy of the array.
- MapWritable and SortedMapWritable are implementations of java.util.Map<Writable, Writable> and java.util.SortedMap<WritableComparable, Writable>, respectively. The type of each key and value field is a part of the serialization format for that field.

3.5.6 Avro

- Data serialization is a technique of converting data into binary or text format. There are multiple systems available for this purpose. Apache Avro is one of those data serialization systems.
- Avro is a language-independent serialization library. Avro is a language independent, schema-based data serialization library. It uses a schema to perform serialization and deserialization. Moreover, Avro uses a JSON format to specify the data structure which makes it more powerful.
- Avro creates a data file where it keeps data along with schema in its metadata section. Avro files include markers that can be used to split large data sets into subsets suitable for Apache MapReduce processing.
- Avro has rich schema resolution capabilities. Within certain carefully defined constraints, the schema used to read data need not be identical to the schema that was used to write the data.
- An Avro data file has a metadata section where the schema is stored, which makes the file self-describing. Avro data files support compression and are splittable, which is crucial for a MapReduce data input format.
- Avro defines a small number of data types, which can be used to build application specific data structures by writing schemas.

- Avro supports two types of data :
  a) **Primitive type :** Avro supports all the primitive types. We use primitive type names to define a type of a given field. For example, a value which holds a string should be declared as {"type": "string"} in Schema.
  b) **Complex type :** Avro supports six kinds of complex types : records, enums, arrays, maps, unions and fixed.

Avro data files :

- A data file has a header containing metadata, including the Avro schema and a sync marker, followed by a series of blocks containing the serialized Avro objects.
- Blocks are separated by a sync marker that is unique to the file and that permits rapid resynchronization with a block boundary after seeking to an arbitrary point in the file, such as an HDFS block boundary. Thus, Avro data files are splittable, which makes them amenable to efficient MapReduce processing.

3.6 File-based Data Structures

- Apache Hadoop supports text files which are quite commonly used for storing the data, besides text files it also supports binary files and one of these binary formats is called **sequence files.**
- Hadoop sequence file is a flat file structure which consists of serialized key-value pairs. This is the same format in which the data is stored internally during the processing of the MapReduce tasks.
- Sequence files can also be compressed for space considerations and based on these compression type users, Hadoop sequence files can be of three types : Uncompressed, record compressed and block compressed.
- To create a SequenceFile, use one of its createWriter() static methods, which returns a SequenceFile.Writer instance.
- The keys and values stored in a SequenceFile do not necessarily need to be writable. Any types that can be serialized and deserialized by a serialization may be used.
- **Reading sequence files** from beginning to end is a matter of creating an instance of **SequenceFile.Reader** and iterating over records by repeatedly invoking one of the next() methods.

The SequenceFile format

- A sequence file consists of a header followed by one or more records. Fig. 3.6.1 shows the internal structure of a sequence file with no compression and record compression.

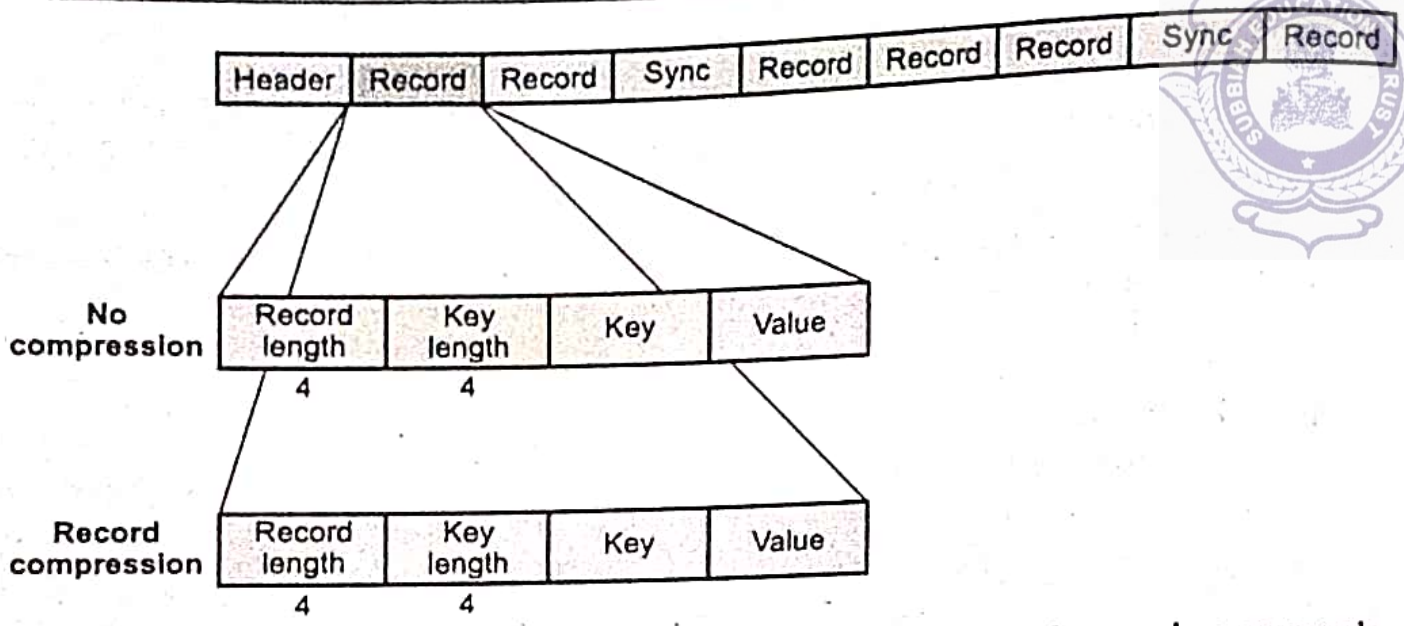


Fig. 3.6.1 Structure of a sequence file with no compression and record compression

- The first three bytes of a sequence file are the bytes SEQ, which acts a magic number, followed by a single byte representing the version number. The header contains other fields including the names of the key and value classes, compression details, user defined metadata and the sync marker.
- Recall that the sync marker is used to allow a reader to synchronize to a record boundary from any position in the file. Each file has a randomly generated sync marker, whose value is stored in the header. Sync markers appear between records in the sequence file.

3.7 Cassandra - Hadoop Integration

- Cassandra provides native support to Hadoop MapReduce, Pig and Hive. Cassandra supports input to Hadoop with ColumnFamilyInputFormat and output with ColumnFamilyOutputFormat classes, respectively.
- ***ColumnFamilyInputFormat :*** It is an implementation of org.apache.hadoop.mapred.InputFormat. So, its implementation is dictated by the InputFormat class specifications. Hadoop uses this class to get data for the MapReduce tasks. It also fragments input data into small chunks that get fed to map tasks.
- ***ColumnFamilyOutputFormat :*** OutputFormat is the mechanism of writing the result from MapReduce to a permanent storage. Cassandra implements Hadoop's OutputFormat. It enables Hadoop to write the result from the reduced task as column family rows. It is implemented such that the results are written, to the column family, in batches. This is a performance improvement and this mechanism is called **lazy write-back caching.**

- *ConfigHelper* : ConfigHelper is a gateway to configure Cassandra-specific settings for Hadoop. It saves developers from inputting a wrong property name because all the properties are set using a method; any typo will appear at compile time.
- *Bulk loading* : BulkOutputFormat is another utility that Cassandra provides to improve the write performance of jobs that result in large data. It streams the data in binary format, which is much quicker than inserting one by one. It uses SSTableLoader to do this.
- Configuring Hadoop with Cassandra is itself quite some work. Writing verbose and long Java code to do something as trivial as word count is a turn-off to a high level user like a data analyst.

3.8 Two Marks Questions with Answers

Q.1 Why do we need Hadoop streaming ?

Ans. : It helps in real-time data analysis, which is much faster using MapReduce programming running on a multi-node cluster. There are different technologies like spark Kafka and others which help in real-time Hadoop streaming.

Q.2 What is the Hadoop Distributed file system ?

Ans. : The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably and to stream those data sets at high bandwidth to user applications. HDFS stores file system metadata and application data separately. The HDFS namespace is a hierarchy of files and directories. Files and directories are represented on the NameNode by inodes, which record attributes like permissions, modification and access times, namespace and disk space quotas.

Q.3 What is data locality optimization ?

Ans. : To run the map task on a node where the input data resides in HDFS. This is called data locality optimization.

Q.4 Why do map tasks write their output to the local disk, not to HDFS ?

Ans. : Map output is intermediate output : It is processed by reducing tasks to produce the final output and once the job is complete the map output can be thrown away. So storing it in HDFS, with replication, would be overkill. If the node running the map task fails before the map output has been consumed by the reduce task, then Hadoop will automatically rerun the map task on another node to re-create the map output.

Q.5 Why is a block in HDFS so large ?

Ans. : HDFS blocks are large compared to disk blocks and the reason is to minimize the cost of seeks. By making a block large enough, the time to transfer the data from the disk can be made to be significantly larger than the time to seek to the start of the block. Thus the time to transfer a large file made of multiple blocks operates at the disk transfer rate.

Q.6 How HDFS services support big data ?

Ans. : Five core elements of big data organized by HDFS services :

- Velocity - How fast data is generated, collated and analyzed.
- Volume - The amount of data generated.
- Variety - The type of data, this can be structured, unstructured, etc.
- Veracity - The quality and accuracy of the data.
- Value - How you can use this data to bring an insight into your business processes.

Q.7 What if writable were not there in Hadoop ?

Ans. : Serialization is important in Hadoop because it enables easy transfer of data. If Writable is not present in Hadoop, then it uses the serialization of Java which increases the data over-head in the network.

Q.8 Define serialization.

Ans. : Serialization is the process of converting object data into byte stream data for transmission over a network across different nodes in a cluster or for persistent data storage.

Q.9 What is writables ? Explain its importance in Hadoop.

Ans. : Writable is an interface in Hadoop. Writable in Hadoop acts as a wrapper class to almost all the primitive data type of Java. That is how int of java has become IntWritable in Hadoop and String of Java has become Text in Hadoop. Writables are used for creating serialized data types in Hadoop. So, let us start by understanding what are data type, interface and serialization.

Q.10 What happens if a client detects an error when reading a block in Hadoop ?

Ans. : If a client detects an error when reading a block :

- It reports the bad block and datanode it was trying to read from to the namenode before throwing a ChecksumException.
- The namenode marks the block replica as corrupt, so it does not direct clients to it, or try to copy this replica to another datanode.
- It then schedules a copy of the block to be replicated on another datanode, so its replication factor is back at the expected level.

- Once this has happened, the corrupt replica is deleted.

Q.11 What is MapFile ?

Ans. : A MapFile is a sorted sequence file with an index to permit lookups by key. Map File can be thought of as a persistent form of java.util. Map which is able to grow beyond the size of a map that is kept in memory.

Q.12 What are Hadoop pipes ?

Ans. : Hadoop pipes is the name of the C++ interface to Hadoop MapReduce. Unlike streaming, this uses standard input and output to communicate with the map and reduce code. Pipes uses sockets as the channel over which the task tracker communicates with the process running the C++ map or reduce function.

steps for integrating cassandra.

1. Setup hadoop cluster.
2. Install apache cassandra
3. Install Hadoop - cassandra connectors.
4. configure the connectors.
5. import data from cassandra to hadoop.
6. perform Hadoop data processing
7. Export Result back to Cassandra.

❑❑❑

UNIT IV

4 Map Reduce Applications

Syllabus

MapReduce workflows - unit tests with MRUnit - test data and local tests - anatomy of MapReduce job run - classic Map-reduce - YARN - failures in classic Map-reduce and YARN - job scheduling - shuffle and sort - task execution - MapReduce types - input formats - output formats.

Contents

4.1 Introduction to MapReduce

- MapReduce is a Java - based, distributed execution framework within the Apache Hadoop Ecosystem. It takes away the complexity of distributed programming by exposing two processing steps that developers implement : Map and Reduce.
- In the Mapping step, data is split between parallel processing tasks. Transformation logic can be applied to each chunk of data. Once completed, the Reduce phase takes over to handle aggregating data from the Map set.
- In general, MapReduce uses Hadoop Distributed File System (HDFS) for both input and output. The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster - node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.
- MapReduce is a programming model and software framework first developed by Google. Intended to facilitate and simplify the processing of vast amounts of data in parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner.

Characteristics of MapReduce :

1. Very large scale data : peta, exa bytes.
2. Write once and read many data. It allows for parallelism without mutexes.
3. Map and Reduce are the main operations : Simple code.
4. All the map should be completed before reduce operation starts.
5. Map and reduce operations are typically performed by the same physical processor.
6. Number of map tasks and reduce tasks are configurable.

4.1.1 MapReduce Workflows

- With HDFS, we are able to distribute the data so that data is stored on hundreds of nodes instead of a single large machine.
- Mapreduce provides the framework for highly parallel processing of data across clusters of commodity hardware. Fig. 4.1.1 shows MapReduce data processing.
- It removes the complicated programming part from the programmers and moves into the framework. Programmers can write simple programs to make use of the parallel processing.

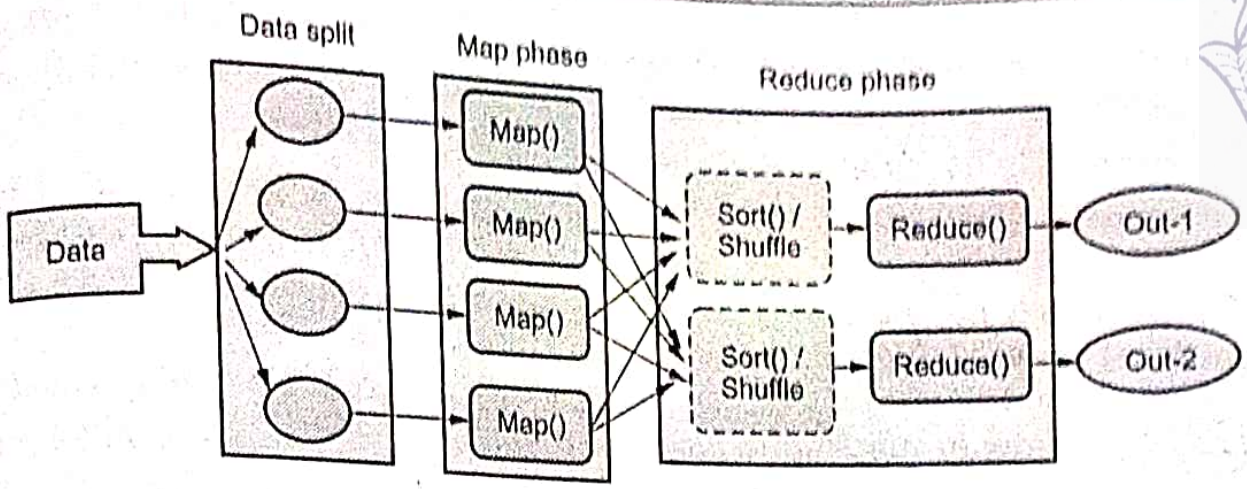


Fig. 4.1.1 MapReduce data processing

- The framework splits the data into smaller chunks that are processed in parallel on cluster of machines by programs called mappers.
- The output from the mappers is then consolidated by reducers into desired result. The share nothing architecture of mappers and reducers make them highly parallel.
- Input : This is the input data / file to be processed.
- Split : Hadoop splits the incoming data into smaller pieces called "splits".
- Map : In this step, MapReduce processes each split according to the logic defined in map() function. Each mapper works on each split at a time. Each mapper is treated as a task and multiple tasks are executed across different TaskTrackers and coordinated by the JobTracker.
- Combine : This is an optional step and is used to improve the performance by reducing the amount of data transferred across the network. Combiner is the same as the reduce step and is used for aggregating the output of the map() function before it is passed to the subsequent steps.
- Shuffle and Sort : In this step, outputs from all the mappers is shuffled, sorted to put them in order and grouped before sending them to the next step.
- Reduce : This step is used to aggregate the outputs of mappers using the reduce() function. Output of reducer is sent to the next and final step. Each reducer is treated as a task and multiple tasks are executed across different TaskTrackers and coordinated by the JobTracker.
- Output : Finally the output of reduce step is written to a file in HDFS.
- The output from the mappers is then consolidated by reducers into desired result. The share nothing architecture of mappers and reducers make them highly parallel.

- Data locality is achieved by mapreduce by working closely with HDFS. When you specify the file system as HDFS for mapreduce, it automatically schedules the mappers on the same node as where the block of data exists.
- Mapreduce can get the blocks from HDFS and process them. The final output from Mapreduce also can be stored in HDFS file system. However, the intermediate files between mappers and reducers are not stored in HDFS and are stored on the local file system of the mappers.

4.1.2 Data Flow in the MapReduce Programming Model

MapReduce : A programming model to facilitate the development and execution of distributed tasks.

- The programmer defines the program logic as two functions :
  a. Map transforms the input into key-value pairs to process.
  b. Reduce aggregates the list of values for each key.

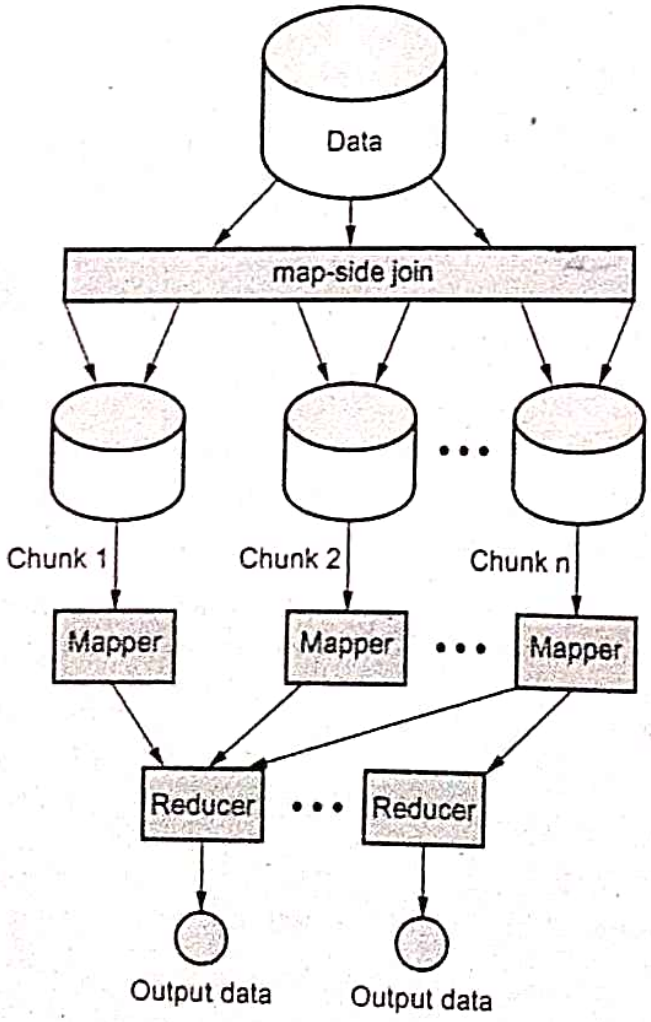


Fig. 4.1.2 Data flow in map reduce

- The MapReduce environment takes in charge distribution aspects. A complex program can be decomposed as a succession of Map and Reduce tasks. Higher-level languages (Pig, Hive, etc.) help with writing distributed applications.
- MapReduce is a parallel programming model especially dedicated for complex and distributed computations, which has been derived from the functional paradigm.
- In general, MapReduce processing is composed of two consecutive stages, which for most problems are repeated iteratively : The map and the reduce phase. Fig. 4.1.2 shows data flow in the MapReduce programming model.
- Map processes the data on hosts in parallel, whereas the reduce aggregates the results. At each iteration independently, the whole data is split into chunks, which, in turn, are used as the input for mappers.
- Each chunk may be processed by only one mapper. Once the data is processed by mappers, they can emit View the MathML source ? key, value ? pairs to the reduce phase.
- Before the reduce phase the pairs are sorted and collected according to the View the MathML sourcekey values, therefore each reducer gets the list of values related to a given View the MathML sourcekey. The consolidated output of the reduce phase is saved into the distributed file system.

4.1.3 Functions of Job Tracker and Task Tracker

Function of Job tracker :

- There is a single job tracker that runs on the master node. It is the driver for the map - reduce jobs. Its functions are :
  1. Accepts jobs from client and divides into tasks.
  2. Schedules tasks on worker nodes called task trackers.
  3. Keeps heartbeat info from task trackers on worker nodes.
  4. Reschedules the task on alternate worker if a worker fails.

Function of task tracker :

- Task tracker runs on each worker node and there are as many task trackers as the worker nodes. If HDFS is also used, then data nodes of HDFS also become worker nodes for task tracker. The functions of a task tracker are :
  a. Takes assignments from job tracker.
  b. Executes the tasks locally.
  c. Each worker node has specific number of mapper and reducer tasks it can take at one time.
  d. The tasks assigned are run in parallel.

e. Normally they can take more map jobs than reduce tasks.

f. Task tracker does a task attempt before executing task.

g. Task tracker may do multiple attempts before declaring a task as failed.

h. Task tracker maintains a connection with the task attempt called umbilical protocol.

i. Task tracker sends a regular heartbeat signal to job tracker indicating its status including available map and reduce tasks.

j. Task tracker runs each task attempt in a separate JVM. So even if the task has bad code due to which it fails, it will not cause task tracker to abort.

4.1.4 Limitation of MapReduce

1. Cannot control the order in which the maps or reductions are run.
2. For maximum parallelism, we need Maps and Reduces to not depend on data generated in the same MapReduce job (i.e. stateless).
3. A database with an index will always be faster than a MapReduce job on unindexed data.
4. Reduce operations do not take place until all Maps are complete.
5. General assumption that the output of Reduce is smaller than the input to Map; large data source used to generate smaller final values.

4.2 Unit Tests with MRUnit

- MRUnit is a JUnit-based Java library that allows us to unit test Hadoop MapReduce programs. There is specialized test suite for testing mapreduce jobs, known as MRUnit.
- MRUnit removes as much of the Hadoop framework as possible while developing and testing. The focus is narrowed to the map and reduce code, their inputs and expected outputs. With MRUnit, developing and testing MapReduce code can be done entirely in the IDE and these tests take fractions of a second to run.
- MRUnit is built on top of the popular JUnit testing framework. It uses the object - mocking library, Mockito, to mock most of the essential Hadoop objects so the user only needs to focus on the map and reduce logic.
- MRUnit supports testing Mappers and Reducers separately as well as testing MapReduce computations as a whole.
- To get started, download MRUnit. After we have extracted the tar file, cd into the mrunit-0.9.0- incubating/lib directory. In there we should see the following :

mrunit-0.9.0-incubating-hadoop1.jar
mrunit-0.9.0-incubating-hadoop2.jar

- The mrunit-0.9.0-incubating-hadoop1.jar is for MapReduce version 1 of Hadoop and mrunit-0.9.0-incubating-hadoop2.jar is for working the new version of Hadoop's MapReduce.
- Given a MapReduce job that writes to an HBase table called MyTest, which has one column family called CF, the reducer of such a job could look like the following :

```
public class MyReducer extends TableReducer<Text, Text, ImmutableBytesWritable> {
 public static final byte[] CF = "CF".getBytes();
 public static final byte[] QUALIFIER = "CQ-1".getBytes();
 public void reduce(Text key, Iterable<Text> values, Context context) throws
 IOException, InterruptedException {
  //bunch of processing to extract data to be inserted, in our case, lets say we
    are simply
  //appending all the records we receive from the mapper for this particular
  //key and insert one record into HBase
  StringBuffer data = new StringBuffer();
  Put put = new Put(Bytes.toBytes(key.toString()));
  for (Text val : values) {
    data = data.append(val);
  }
  put.add(CF, QUALIFIER, Bytes.toBytes(data.toString()));
  //write to HBase
  context.write(new ImmutableBytesWritable(Bytes.toBytes(key.toString())), put);
 }
}
```

- **To test this code, the first step is to add a dependency to MRUnit to Maven POM file.**

```
<dependency>
 <groupId>org.apache.mrunit</groupId>
 <artifactId>mrunit</artifactId>
 <version>1.0.0 </version>
 <scope>test</scope>
</dependency>
```

- **Next, use the ReducerDriver provided by MRUnit, in Reducer job.**

```
public class MyReducerTest {
  ReduceDriver<Text, Text, ImmutableBytesWritable, Writable> reduceDriver;
  byte[] CF = "CF".getBytes();
  byte[] QUALIFIER = "CQ-1".getBytes();
  @Before
  public void setUp() {
    MyReducer reducer = new MyReducer();
```

```
    reduceDriver = ReduceDriver.newReduceDriver(reducer);
  }
   @Test
 public void testHBaseInsert() throws IOException {
    String strKey = "RowKey-1", strValue = "DATA", strValue1 = "DATA1", strValue2 =
    "DATA2";
    List<Text> list = new ArrayList<Text>();
    list.add(new Text(strValue));
    list.add(new Text(strValue1));
    list.add(new Text(strValue2));
    //since the reducer is doing is appending the records that the mapper
    //sends it, we should get the following back
    String expectedOutput = strValue + strValue1 + strValue2;
   //Setup Input, mimic what mapper would have passed
    //to the reducer and run test
    reduceDriver.withInput(new Text(strKey), list);
    //run the reducer and get its output
    List<Pair<ImmutableBytesWritable, Writable>> result = reduceDriver.run();
       //extract key from result and verify
    assertEquals(Bytes.toString(result.get(0).getFirst().get()), strKey);
       //extract value for CF/QUALIFIER and verify
    Put a = (Put)result.get(0).getSecond();
    String c = Bytes.toString(a.get(CF, QUALIFIER).get(0).getValue());
    assertEquals(expectedOutput,c );
  }
}
```

- MRUnit test verifies that the output is as expected, the Put that is inserted into HBase has the correct value and the ColumnFamily and ColumnQualifier have the correct values. MRUnit includes a MapperDriver to test mapping jobs and we can use MRUnit to test other operations, including reading from HBase, processing data, or writing to HDFS.
- 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 and environment setup
  5. Write a method to test mapper
  6. Write a method to test reducer
  7. Write a method to test the whole MapReduce job
  8. Run the test
- How to test Java MapReduce Jobs in Hadoop ?

Developer activities :

Step 1 : Develop MapReduce Code

Step 2 : Unit Testing of Map Reduce code using MRUnit framework

Step 3 : Create Jar file for MapReduce code

Testing activities :

Step 1 : Create a new directory in HDFS then copy data file from local to HDFS directory

Step 2 : Run jar file by providing data file as an input

Step 3 : Check output file created on HDFS.

4.3 Anatomy of MapReduce Job Run

- We can run a MapReduce job with a single line of code : JobClient.runJob(conf). This single line code execution process is shown below.

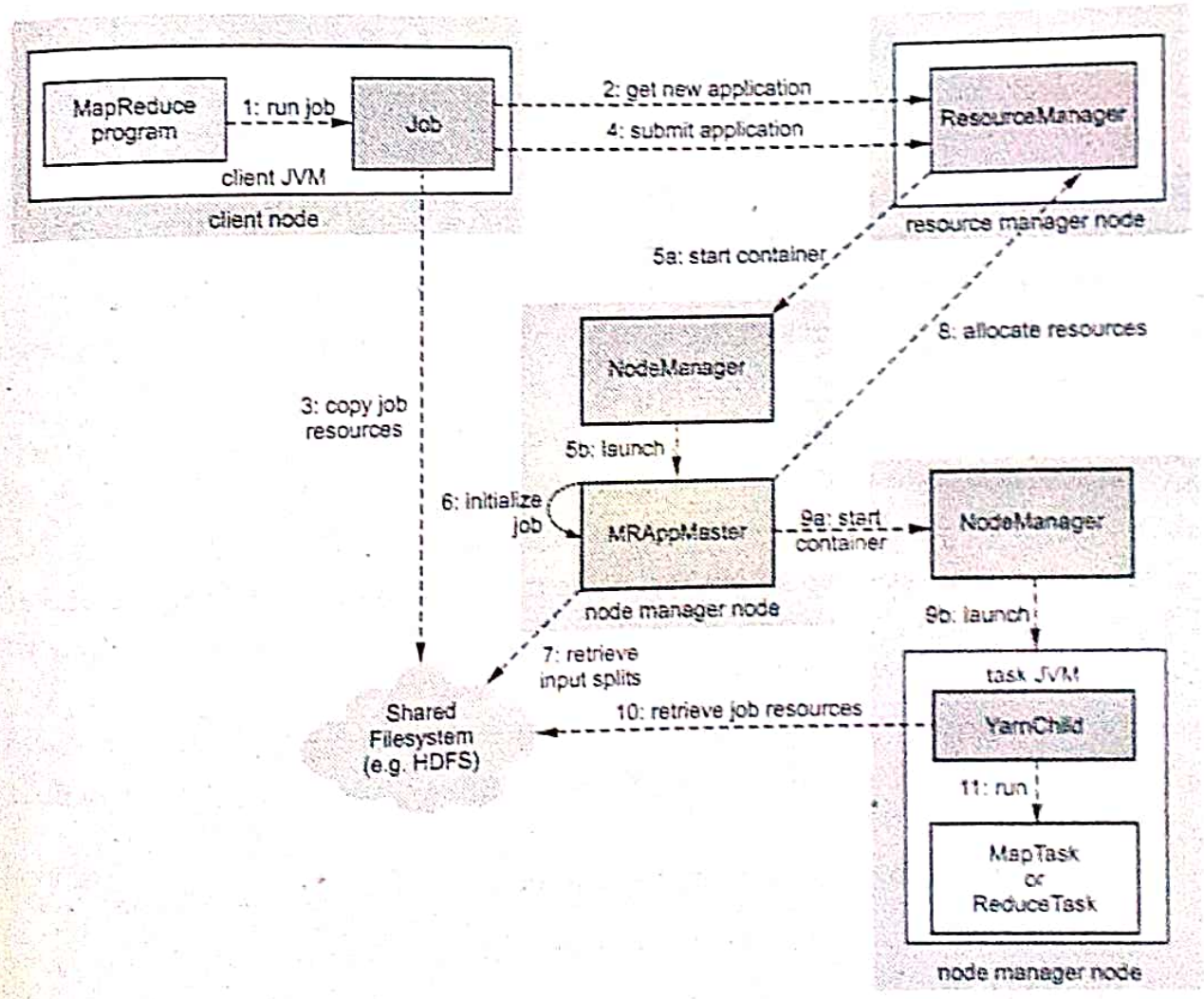


Fig. 4.3.1 How Hadoop runs a MapReduce job

- There are five independent entities :
  1. The client, which submits the MapReduce job.
  2. The YARN resource manager, which coordinates the allocation of compute resources on the cluster.
  3. The YARN node managers, which launch and monitor the compute containers on machines in the cluster.
  4. The MapReduce application master, which coordinates the tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager and managed by the node managers.
  5. The distributed file system, which is used for sharing job files between the other entities.

Job submission :

1. The submit() method on Job creates an internal JobSubmitter instance and calls submitJobInternal() on it.
2. Having submitted the job, waitForCompletion polls the job's progress once per second and reports the progress to the console if it has changed since the last report.
3. When the job completes successfully, the job counters are displayed otherwise, the error that caused the job to fail is logged to the console.

- The job submission process implemented by JobSubmitter does the following :
  1. Asks the resource manager for a new application ID, used for the MapReduce job ID.
  2. Checks the output specification of the job For example, if the output directory has not been specified or it already exists, the job is not submitted and an error is thrown to the MapReduce program.
  3. Computes the input splits for the job If the splits cannot be computed (because the input paths don't exist, for example), the job is not submitted and an error is thrown to the MapReduce program.
  4. Copies the resources needed to run the job, including the job JAR file, the configuration file and the computed input splits, to the shared filesystem in a directory named after the job ID.
  5. Submits the job by calling submitApplication() on the resource manager.

Job initialization :

1. When the resource manager receives a call to its submitApplication() method, it hands off the request to the YARN scheduler.
2. The scheduler allocates a container and the resource manager then launches the application master's process there, under the node manager's management.
3. The application master for MapReduce jobs is a Java application whose main class is MRAppMaster.
4. It initializes the job by creating a number of bookkeeping objects to keep track of the job's progress, as it will receive progress and completion reports from the tasks.
5. It retrieves the input splits computed in the client from the shared filesystem.
6. It then creates a map task object for each split, as well as a number of reduce task objects determined by the mapreduce.job.reduces property (set by the setNumReduceTasks() method on Job).

Task assignment :

1. If the job does not qualify for running as an uber task, then the application master requests containers for all the map and reduce tasks in the job from the resource manager.
2. Requests for map tasks are made first and with a higher priority than those for reduce tasks, since all the map tasks must complete before the sort phase of the reduce can start.
3. Requests for reduce tasks are not made until 5 % of map tasks have completed.

Task execution :

1. Once a task has been assigned resources for a container on a particular node by the resource manager's scheduler, the application master starts the container by contacting the node manager.
2. The task is executed by a Java application whose main class is YarnChild. Before it can run the task, it localizes the resources that the task needs, including the job configuration and JAR file and any files from the distributed cache.
3. Finally, it runs the map or reduce task.

Streaming :

- Streaming runs special map and reduce tasks for the purpose of launching the user supplied executable and communicating with it. The Streaming task communicates with the process using standard input and output streams.

- During execution of the task, the Java process passes input key value pairs to the external process, which runs it through the user defined map or reduce function and passes the output key value pairs back to the Java process.
- From the node manager's point of view, it is as if the child process ran the map or reduce code itself.

Progress and status updates :

- MapReduce jobs are long running batch jobs, taking anything from tens of seconds to hours to run.
- A job and each of its tasks have a status, which includes such things as the state of the job or task, the progress of maps and reduces, the values of the job's counters and a status message or description.
- When a task is running, it keeps track of its progress. For map tasks, this is the proportion of the input that has been processed. For reduce tasks, it's a little more complex, but the system can still estimate the proportion of the reduce input processed.

Job completion :

- When the application master receives a notification that the last task for a job is complete, it changes the status for the job to Successful. Then, when the Job polls for status, it learns that the job has completed successfully, so it prints a message to tell the user and then returns from the waitForCompletion().
- Finally, on job completion, the application master and the task containers clean up their working state and the OutputCommitter's commitJob () method is called.
- Job information is archived by the job history server to enable later interrogation by users if desired.

4.4 YARN

- YARN stands for Yet Another Resource Negotiator. It is the next generation computing framework in Apache Hadoop with support for programming paradigms besides MapReduce. It is a large - scale distributed operating system for big data applications.
- It has two major responsibilities :
  1. Management of cluster resources such as compute, network and memory.
  2. Scheduling and monitoring of jobs.
- YARN in Hadoop allows for the execution of various data processing engines such as batch processing, graph processing, stream processing and interactive processing, as well as the processing of data stored in HDFS.

- Why is YARN Used ?
  a) YARN in Hadoop efficiently and dynamically allocates all cluster resources, resulting in higher Hadoop utilization compared to previous versions which help in better cluster utilization.
  b) Clusters in YARN in Hadoop can now run streaming data processing and interactive queries in parallel with MapReduce batch jobs.
  c) It can now handle several processing methods and can support a wider range of applications.
- Fig. 4.4.1 shows YARN architecture. The architecture consists of several components such as Resource Manager, Node Manager and Application Master.

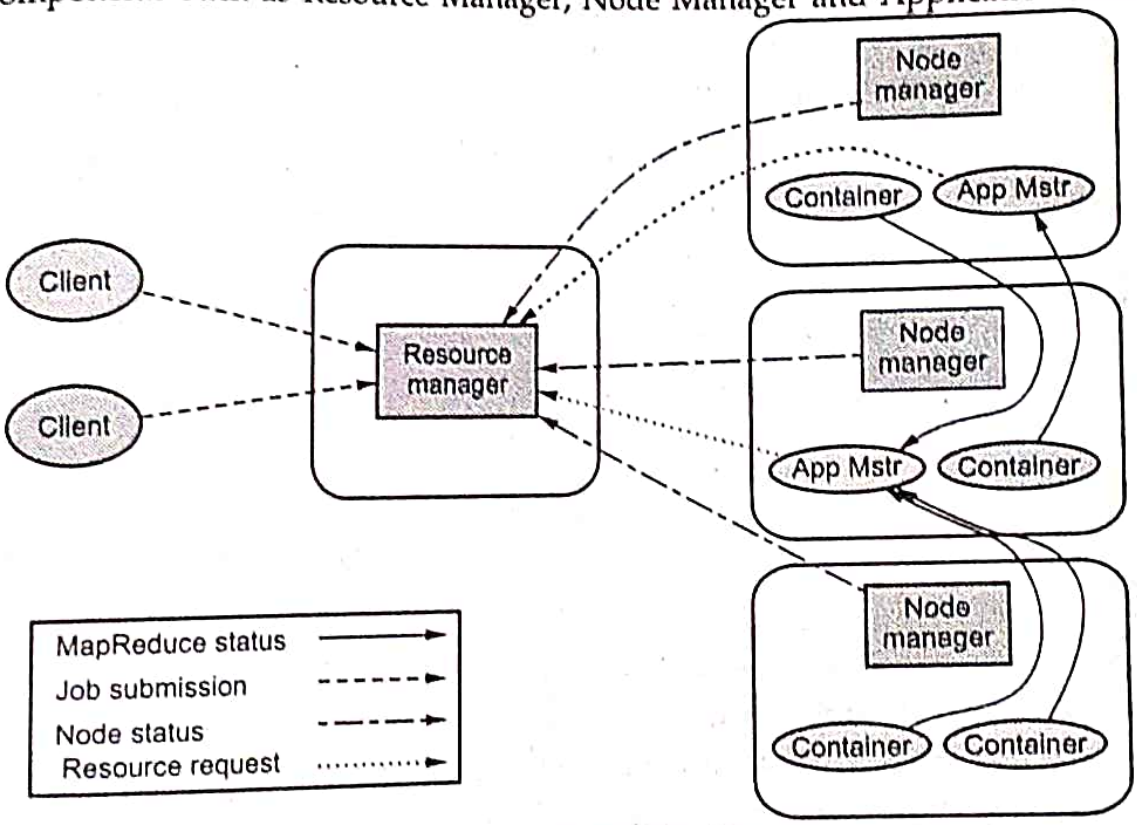


Fig. 4.4.1 YARN architecture

- YARN Architecture consists of the following main components :
  a) Resource manager : Runs on a master daemon and manages the resource allocation in the cluster.
  b) Node manager : They run on the slave daemons and are responsible for the execution of a task on every single data node.
  c) Application master : Manages the user job lifecycle and resource needs of individual applications. It works along with the node manager and monitors the execution of tasks.
  d) Container : Package of resources including RAM, CPU, Network, HDD etc on a single node.

- The resource manager and the node manager form the data-computation framework. The resource manager is the ultimate authority that arbitrates resources among all the applications in the system. The node manager is the per-machine framework agent who is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the resource manager/scheduler.
- YARN containers are managed by a container launch context which is Container Life-Cycle(CLC). This record contains a map of environment variables, dependencies stored in a remotely accessible storage, security tokens, payload for node manager services and the command necessary to create the process.
- Application workflow in YARN :
  a) Client submits an application
  b) Resource manager allocates a container to start application manager
  c) Application manager registers with resource manager
  d) Application Manager asks containers from resource manager
  e) Application Manager notifies node manager to launch containers
  f) Application code is executed in the container
  g) Client contacts resource manager/application manager to monitor application's status
  h) Application manager unregisters with resource manager

Architecture features of YARN :

- YARN has become popular for the following reasons -
  a) With the scalability of the resource manager of the YARN architecture, Hadoop may manage thousands of nodes and clusters.
  b) YARN compatibility with Hadoop 1.0 is maintained by not affecting map-reduce applications.
  c) Dynamic utilization of clusters in Hadoop is facilitated by YARN, which gives better cluster utilization.
  d) Multi-tenancy enables an organization to gain the benefits of multiple engines at once.

4.4.1 Merits and Demerits of YARN

1. Merits :

- Scalability : YARN is designed for large number of nodes.
- Utilization : Node manager manages a pool of resources, rather than a fixed number of the designated slots thus increasing the utilization.

- Multitenancy : Different version of MapReduce can run on YARN, which makes the process of upgrading MapReduce more manageable.

2. Demerit

- Availability - JobTracker is the only point of availability in Hadoop 1.0

4.4.2 Difference between YARN and MapReduce

YARN	MapReduce
Used in Hadoop version 2	Used in Hadoop version 1
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.
HADOOP2, based on YARN architecture, has the concept of multiple masters and slaves	Map reduce has a single master and multiple slave architecture
Default node size of Datanode is 128 MB	Default node size of Datanode is 64 MB
YARN is more isolated and scalable	MapReduce is less scalable than YARN.
YARN can dynamically allocate pool of resources to applications	Provided static allocations of resources for designated work
Supports variety of processing engines and applications.	Supported its own batch processing applications only.

4.5 Failures in Classic Map Reduce and YARN

1. Failures in classic MapReduce

- MapReduce support three types of failures :
  a) Running task failure
  b) Tasktracker failure
  c) Jobtracker failure

Task failure

- Child task fail : This happens when user code in the map or reduce task throws a runtime exception. If this happens, the child JVM reports the error back to its parent tasktracker, before it exits. The error ultimately makes it into the user logs. The tasktracker marks the task attempt as failed, freeing up a slot to run another task.

- Streaming tasks fail : if the streaming process exits with a nonzero exit code, it is marked as failed. This behavior is governed by the stream.non.zero.exit.is.failure property.
- The tasktracker notices that it hasn't received a progress update for a while and proceeds to mark the task as failed. The child JVM process will be automatically killed after this period.

Tasktracker failure

- If a tasktracker fails by crashing, or running very slowly, it will stop sending heartbeats to the jobtracker. The jobtracker will notice a tasktracker that has stopped sending heartbeats and remove it from its pool of tasktrackers to schedule tasks on.
- The jobtracker arranges for map tasks that were run and completed successfully on that tasktracker to be rerun if they belong to incomplete jobs, since their intermediate output residing on the failed tasktracker's local filesystem may not be accessible to the reduce task. Any tasks in progress are also rescheduled.
- A tasktracker can also be blacklisted by the jobtracker, even if the tasktracker has not failed. If more than four tasks from the same job fail on a particular tasktracker, then the jobtracker records this as a fault.

Jobtracker failure

- It is most serious failure mode. It is a single point of failure. Hadoop has no mechanism for dealing with failure of the jobtracker.
- YARN is used to overcome this situation.
- After restarting a jobtracker, any jobs that were running at the time it was stopped will need to be re - submitted.

2. Failures in YARN

- **Task failure :** Failure of the running task is similar to the classic case. Runtime exceptions and sudden exits of the JVM are propagated back to the application master and the task attempt is marked as failed.
- **Node manager failure :** If a node manager fails, then it will stop sending heartbeats to the resource manager and the node manager will be removed from the resource manager's pool of available nodes. Any task or application master running on the failed node manager will be recovered using the mechanisms.
- **Resource manager failure :** Failure of the resource manager is serious, since without it neither jobs nor task containers can be launched. The resource manager was designed from the outset to be able to recover from crashes, by using a checkpointing mechanism to save its state to persistent storage, although at the time of writing the latest release did not have a complete implementation.

4.6 Job Scheduling

- The Hadoop schedulers are designed for better utilization of resources and performance enhancement. Requirements (issues and challenges) regarding job scheduling in Hadoop are as follows :
- **Energy efficiency :** To perform operations on large amount of data, a large amount of energy is required in data centers, which increase the overall cost. The minimization of energy in data centers is a big challenge in Hadoop.
- **Load balancing :** Map and Reduce stages are linked with the partition stage. By default, the data is equally portioned by partition algorithms, which handle the system imbalance in case skewed data is encountered. As only the key is considered in processing and not the data size, load balancing problem occurs.
- **Mapping scheme :** Mapping scheme that could be helpful in the minimization of communication cost is required.
- **Automation and configuration :** It helps in the deployment of Hadoop cluster by setting all the parameters. For proper configuration, both the hardware and work amount should be known at the timing of deployment. During configuration, a small mistake could cause inefficient execution of the job, leading to performance degradation. To overcome such types of issues, we need to develop new techniques and algorithms that perform calculations in such a manner that the setting could be done efficiently.
- **Fairness :** Fairness refer to the fairness in scheduling algorithms. It indicates how fairly of scheduling algorithms divide the resources among users.
- **Data locality :** The distance between the task node and input node is known as locality. The data transfer rate depends on the locality. The data transfer time will be short if the computational node is near the input node.
- **Synchronization :** The process of transferring the intermediate output data of the mapping process as the input data to the reduce process is known as synchronization.

4.6.1 FIFO Scheduler

- Default scheduling policy used in Hadoop is First In First Out. More preferences is given to the application coming first than those coming later. It places the applications in a queue and executes them in the order of their submission (first in, first out).
- Here, irrespective of the size and priority, the request for the first application in the queue are allocated first. Once the first application request is satisfied, then only the next application in the queue is served.

- Advantage of FIFO scheduler :
  1. It is simple to understand and doesn't need any configuration.
  2. Jobs are executed in the order of their submission.
- Disadvantage of FIFO scheduler
  1. It is not suitable for shared clusters.
  2. It does not take into account the balance of resource allocation between the long applications and short applications.

4.6.2 Fair Scheduler

- Fair scheduler is developed Facebook. Fair scheduler aims to give every user a fair share of the cluster capacity over time. If a single job is running, it gets all of the cluster. As more jobs are submitted, free task slots are given to the jobs in such a way as to give each user a fair share of the cluster.
- The main idea behind fair scheduler is to allocate equal share of resources to each job. It creates groups of jobs based on configurable attributes like user name, called pools.
- The fair scheduler ensures fairness in sharing of resources between pools. The pools also control job configurable properties and the configurable properties determine the pool in which a job is placed.
- All the users have their own pools with a minimum share assigned to each. Minimum share means that a small part of the total number of slots is always achieved by a pool.
- By default, there is a fair allocation of resources among the pools with the MapReduce task slot. If any pool is free i.e. they are not being used, then their idle slots will be used by the other pools.
- If the same user or same pool sends too many jobs , then the fair scheduler can limit these jobs by marking the jobs as not runnable. If there is only a single job running at a given time, then it can use the entire cluster.
- Advantages of Fair scheduler
  1. This scheduler makes a fair and dynamic resource reallocation.
  2. It provides faster response to small jobs than large jobs.
  3. It has the ability to fix the number of concurrent running jobs from each user and pool.

- Disadvantages Fair scheduler
  1. Fair scheduler has more complicated configurations.
  2. This scheduler does not consider the weight of each job, which leads to unbalanced performance in each pool/node.
  3. Pools have a limitation on the number of running jobs under fair scheduling.

4.6.3 Capacity Scheduler

- Yahoo developed capacity scheduler. The main objective of this scheduler is to maximize the utilization of resources and throughput in a cluster environment.
- This scheduling algorithm can ensure fair management of computational resources among a large number of users. It uses queues and each queue will be assigned to an organization after the resources have been divided among these queues.
- In order to get control over the queues, a security mechanism is built to ensure that each organization can access only one of the queues. It can never access the queues of another organization.
- This scheduler guarantees minimum capacity by having limits on the running tasks and jobs from a single queue.
- When new jobs arrive in a queue, the resources are assigned back to the previous queue after completion of the currently running jobs. Capacity scheduler allows job scheduling based on priority in an organization's queue.
- Advantages of capacity scheduler
  1. Capacity scheduling policy maximizes utilization of resources and throughput in cluster environment.
  2. This scheduler guarantees the reuse of the unused capacity of the jobs within queues.
  3. It also supports the features of hierarchical queues, elasticity and operability.
  4. It can allocate and control memory based on the available hardware resources.
- Disadvantages of capacity scheduler
  1. Capacity scheduler is the most complex among the other schedulers.
  2. There is difficulty in choosing proper queues.
  3. With regard to pending jobs, it has some limitations in ensuring stability and fairness of the cluster from a queue and single user.

4.6.4 Difference between Fair and Capacity Scheduler

Fair scheduler	Capacity scheduler
Fair scheduler is developed Facebook.	Yahoo developed capacity scheduler.
Fair scheduler assigns equal amount of resource to all running jobs.	Capacity scheduler assigns resource based on the capacity required by the organization.
It support hierarchical XML configuration.	It cannot support hierarchical XML configuration.
Fair scheduler is not complex.	It is complex amongst the other scheduler.
The main idea behind fair scheduler is to allocate equal share of resources to each job.	The main objective of this scheduler is to maximize the utilization of resources and throughput in a cluster environment.

4.7 Shuffle and Sort

- Map-Reduce gives the guarantee that input to every reducer is sorted by key. The process by which system performs the sort and transfers the map outputs to the reducers as inputs are called shuffle.
- When we run a MapReduce job and mappers start producing output internally lots of processing is done by the Hadoop framework before the reducers get their input. Hadoop framework also guarantees that the map output is sorted by keys. This whole internal processing of sorting map output and transferring it to reducers is known as shuffle phase in Hadoop framework.
- Fig. 4.7.1 shows shuffle and sort.
- It is a phase which happens between each Map and Reduce phase. Just to remind Map and Reduce handles the data which are organised into key - value pairs. Once the Mappers are done with the calculations, the results of each Mapper are sorted by the key in so called buffers.
- Once the buffers are filled up, the data spills on the machines' local disks. Then it is straightly pushed to the Reducers (the shuffle phase), so that records associated with the same key end up in the same Reducer.
- This shuffling happens as soon as each Mapper finishes in order not to over-flood the network, which could happen if we waited for all of the Mappers to finish.
- Depending on the setup Reducers may start running before all Mappers have finished their jobs. When data gets to Reducers it is sorted once again. Results are then written to the HDFS or any other used filesystem.
- In order to reduce the amount of data shuffled through the network we may define a combiner function, which does the same calculations as Reducer but on the Mappers' side, so before the data is being transferred.

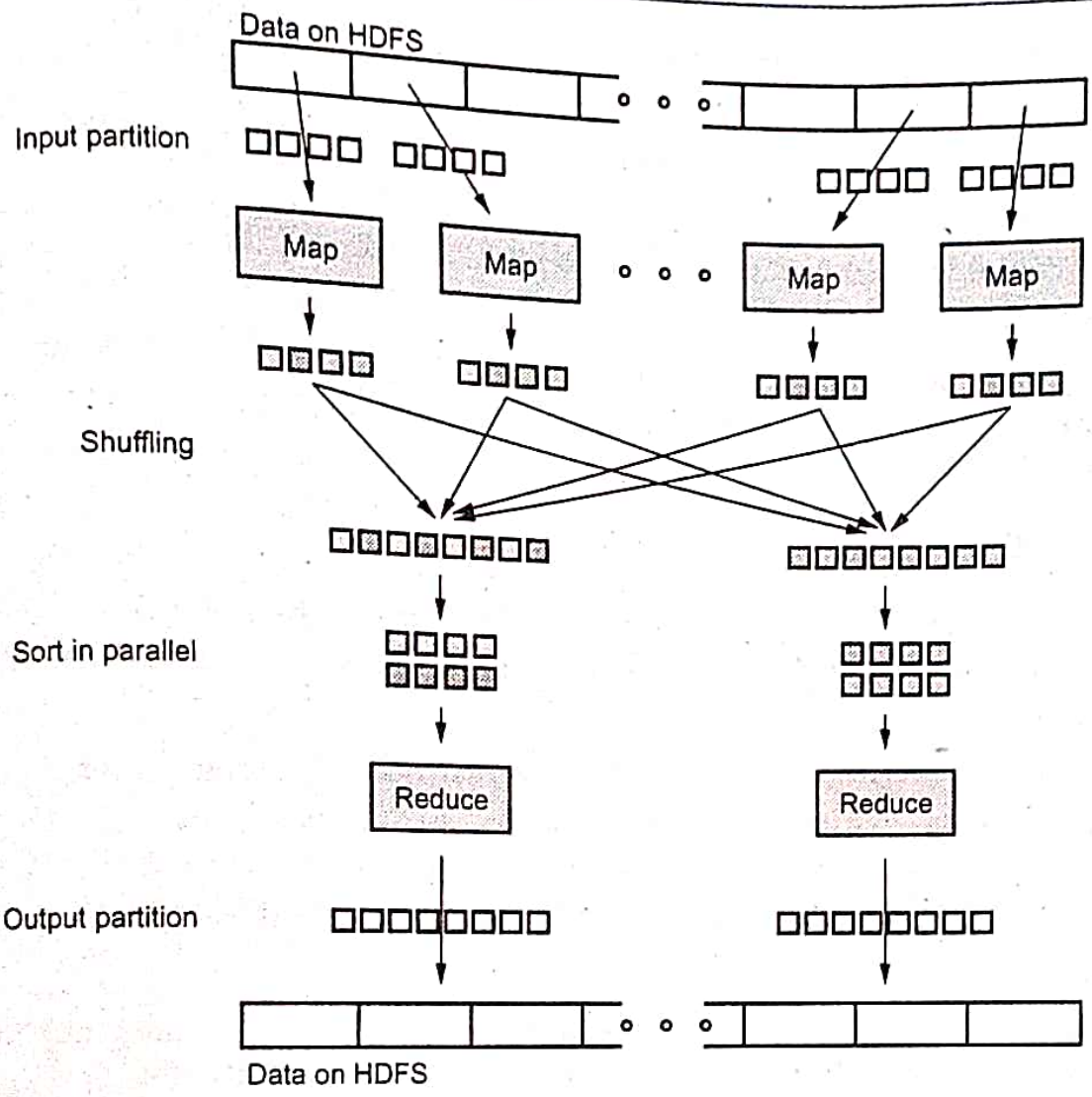


Fig. 4.7.1 Shuffle and sort

- Role of the combiner is to simply pre-calculate the data so that we send less over the network. However there is no way of controlling how many times and if at all combiner will actually be used - Hadoop decides on that.
- Hadoop has a default shuffle and sort mechanism which is based on alphabetical sorting and hash shuffling of the keys. However there is a way of implementing a custom mechanism by overwriting the following classes :
  1. Partitioner - According to which the data will be shuffled.
  2. RawComparator - Responsible for data sorting on the Mapper side.
  3. RawComparator - Which handles the data grouping on the Reducer side.
- The tasks done internally by Hadoop framework with in the shuffle phase are as follows -
  1. Data from mappers is partitioned as per the number of reducers.
  2. Data is also sorted by keys within a partition.
  3. Output from Maps is written to disk as many temporary files.

4. Once the map task is finished all the files written to the disk are merged to create a single file.
5. Data from a particular partition (from all mappers) is transferred to a reducer that is suppose to process that particular partition.
6. If data transferred to a reducer exceeded the memory limit then it is copied to a disk.
7. Once reducer has got its portion of data from all the mappers, data is again merged while still maintaining the sort order of keys to create reduce task input.

4.8 Task Execution

- MapReduce model is to break jobs into tasks and run the tasks in parallel to make the overall job execution time smaller than it would otherwise be if the tasks ran sequentially.
- In Hadoop, speculative execution is a process that takes place during the slower execution of a task at a node. In this process, the master node starts executing another instance of that same task on the other node. And the task which is finished first is accepted and the execution of other is stopped by killing that.

Speculative execution in Hadoop

- Speculative execution in Hadoop MapReduce is an option to run a duplicate map or reduce task for the same input data on an alternative node. This is done so that any slow running task doesn't slow down the whole job.
- MapReduce job is dominated by the slowest task. MapReduce attempts to locate slow tasks, called stragglers. If a straggler is discovered, a redundant (speculative) task is run that will optimistically commit before the corresponding straggler.
- Whichever copy (among the two copies) of a task commits first, it becomes the definitive copy and the other copy is killed by the Jobtracker. This process is known as speculative execution. Only one copy of a straggler is allowed to be speculated.
- **How does Hadoop locate stragglers ?** Hadoop monitors each task progress using a progress score between 0 and 1. If a task's progress score is less than average and the task has run for at least 1 minute, it is marked as a straggler.
- Speculative execution is turned on by default. It can be enabled or disabled independently for map tasks and reduce tasks, on a cluster-wide basis or on a per-job basis.
- Speculative execution is enabled by default for both map and reduce tasks. Properties for speculative execution are set in mapred - site.xml file.

- **Advantages of speculative execution :** In many of the production environments, we can have large-scale clusters with thousands of nodes running MapReduce or the related Hadoop jobs at the same time. Problems like hardware failure and network clusters are common in large - scale clusters. So, it makes sense to run duplicate tasks in case one server might fail.

4.9 MapReduce Types

- The map and reduce functions in Hadoop MapReduce have the following general form :

map: (K1, V1) → list(K2, V2)
reduce: (K2, list(V2)) → list(K3, V3)

- In general, the map input key and value types (K1 and V1) are different from the map output types (K2 and V2). However, the reduce input must have the same types as the map output, although the reduce output types may be different again (K3 and V3).

4.9.1 Input Formats - Output

1. Input formats

- Hadoop can process many different types of data formats, from flat text files to databases.
- An input split is a chunk of the input that is processed by a single map. Each map processes a single split. Each split is divided into records and the map processes each record, a key-value pair.
- Splits and records are logical : There is nothing that requires them to be tied to files. In a database context, a split might correspond to a range of rows from a table and a record to a row in that range. Input splits are represented by the Java interface i.e. InputSplit.
- An InputSplit has a length in bytes and a set of storage locations, which are just hostname strings. An InputFormat is responsible for creating the input splits and dividing them into records.
- FileInputFormat : FileInputFormat is the base class for all implementations of InputFormat that use files as their data source. It provides two things : A place to define which files are included as the input to a job and an implementation for generating splits for the input files. The job of dividing splits into records is performed by subclasses.
- **Small files and CombineFileInputFormat :** Hadoop works better with a small number of large files than a large number of small files. One reason for this is that FileInputFormat generates splits in such a way that each split is all or part of a

single file. If the file is very small and there are a lot of them, then each map task will process very little input and there will be a lot of them (one per file), each of which imposes extra bookkeeping overhead.

- An InputFormat object creates the input splits, that is delegated by the context to get the records. It is the public and customizable run() method of the Mapper to get the records from context.
- FileInputFormat and DBInputFormat classes are derived from InputFormat. The FileInputFormat is further specialized, with classes that i.e. combine small files or prevent file splitting

2. Text Input

- TextInputFormat : This file format is the default InputFormat. Each record is a line of input. The key, a LongWritable, is the byte offset within the file of the beginning of the line. The value is the contents of the line, excluding any line terminators (newline, carriage return) and is packaged as a Text object.
- TextInputFormat is the default InputFormat. Each record is a line of input. The key, a LongWritable, is the byte offset within the file of the beginning of the line. The value is the contents of the line, excluding any line terminators (e.g., newline or carriage return) and is packaged as a Text object.
- A file is broken into splits at byte, not line, boundaries. Splits are processed independently.

Relationship between Input splits and HDFS blocks :

- The logical records that FileInputFormats define usually do not fit neatly into HDFS blocks. For example, a TextInputFormat's logical records are lines, which will cross HDFS boundaries more often than not.
- This has no bearing on the functioning of our program : Lines are not missed or broken
- This means it could be needed to perform some remote reads. The slight overhead this causes is not normally significant. Fig. 4.9.1 shows logical records and HDFS blocks for TextInputFormat
- A single file is broken into lines and the line boundaries do not correspond with the HDFS block boundaries. Splits honor logical record boundaries, in this case lines, so we see that the first split contains line 5, even though it spans the first and second block. The second split starts at line 6.

Binary Input :

- Hadoop MapReduce is not restricted to processing textual data. It has support for binary formats, too.

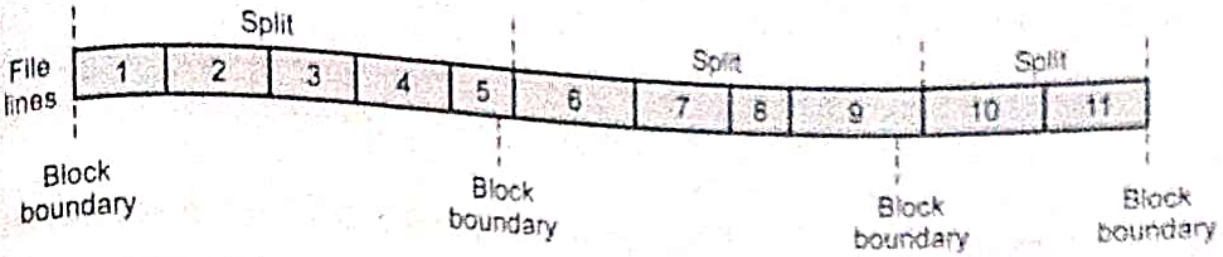


Fig. 4.9.1 Logical records and HDFS blocks for TextInputFormat

- Hadoop's SequenceFileInputFormat stores sequences of binary key - value pairs. Sequence files are splittable, they support compression as a part of the format and they can store arbitrary types.
- SequenceFileAsBinaryInputFormat is a variant of SequenceFileInputFormat that retrieves the sequence file's keys and values as opaque binary objects. They are encapsulated as BytesWritable objects and the application is free to interpret the underlying byte array.

4.10 Two Marks Questions with Answers

Q.1 Define MapReduce.

Ans. : MapReduce is a programming model and software framework first developed by Google. Intended to facilitate and simplify the processing of vast amounts of data in parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner.

Q.2 List the characteristics of MapReduce ?

Ans. : Characteristics of MapReduce

1. Very large scale data : peta, exa bytes.
2. Write once and read many data. It allows for parallelism without mutexes.
3. Map and Reduce are the main operations : Simple code.
4. All the map should be completed before reduce operation starts.
5. Map and reduce operations are typically performed by the same physical processor.
6. Number of map tasks and reduce tasks are configurable.
7. Operations are provisioned near the data.
8. Commodity hardware and storage.

Q.3 What are the major responsibilities of YARN ?

Ans. : It has two major responsibilities :

- Management of cluster resources such as compute, network and memory.
- Scheduling and monitoring of jobs.

Q.4 Why Is YARN Used ?

Ans. :

a) YARN in Hadoop efficiently and dynamically allocates all cluster resources resulting in higher Hadoop utilization compared to previous versions which help in better cluster utilization.

b) Clusters in YARN in Hadoop can now run streaming data processing and interactive queries in parallel with MapReduce batch jobs.

c) It can now handle several processing methods and can support a wider range of applications.

Q.5 What is fair scheduler ?

Ans. : Fair scheduler aims to give every user a fair share of the cluster capacity over time. If a single job is running, it gets all of the cluster. As more jobs are submitted, free task slots are given to the jobs in such a way as to give each user a fair share of the cluster

Q.6 List the failures of MapReduce.

Ans. : MapReduce support three types of failures : Running task failure, Tasktracker failure and Jobtracker failure.

Q.7 Explain First In First Out (FIFO) scheduling.

Ans. : FIFO scheduling policy gives more preference to the jobs coming in earlier than those coming in later. When new jobs arrive, the Job Tracker pulls the earliest job first from the queue.

Q.8 Why Hadoop works better with a small number of large files ?

Ans. : Hadoop works better with a small number of large files than a large number of small files. One reason for this is that FileInputFormat generates splits in such a way that each split is all or part of a single file. If the file is very small and there are a lot of them, then each map task will process very little input and there will be a lot of them (one per file), each of which imposes extra bookkeeping overhead.

Q.9 What is TextInputFormat ?

Ans. : TextInputFormat is the default InputFormat. Each record is a line of input. The key, a LongWritable, is the byte offset within the file of the beginning of the line. The value is the contents of the line, excluding any line terminators (e.g., newline or carriage return) and is packaged as a Text object. A file is broken into splits at byte, not line, boundaries. Splits are processed independently.

Q.10 What is Node Manager Failure in YARN ?

Ans. : If a node manager fails, then it will stop sending heartbeats to the resource manager and the node manager will be removed from the resource manager's pool of available nodes. Any task or application master running on the failed node manager will be recovered using the mechanisms.

❑❑❑

UNIT V

5 Hadoop Related Tools

Syllabus

Hbase - data model and implementations - Hbase clients - Hbase examples - praxis. Pig - Grunt - pig data model - Pig Latin - developing and testing Pig Latin scripts. Hive - data types and file formats - HiveQL data definition - HiveQL data manipulation - HiveQL queries.

Contents

5.1 Hbase

- HBase is an open source, non-relational, distributed database modeled after Google's BigTable. HBase is an open source and sorted map data built on Hadoop. It is column oriented and horizontally scalable.
- It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop file system. It runs on top of Hadoop and HDFS, providing Big Table-like capabilities for Hadoop.
- HBase supports massively parallelized processing via MapReduce for using HBase as both source and sink.
- HBase supports an easy-to-use Java API for programmatic access. It also supports Thrift and REST for non-Java front-ends.
- HBase is a column oriented distributed database in Hadoop environment. It can store massive amounts of data from terabytes to petabytes. HBase is scalable, distributed big data storage on top of the Hadoop eco system.
- The HBase physical architecture consists of servers in a Master-Slave relationship. Typically, the HBase cluster has one Master node, called HMaster and multiple Region Servers called HRegionServer. Fig. 5.1.1 shows Hbase architecture.

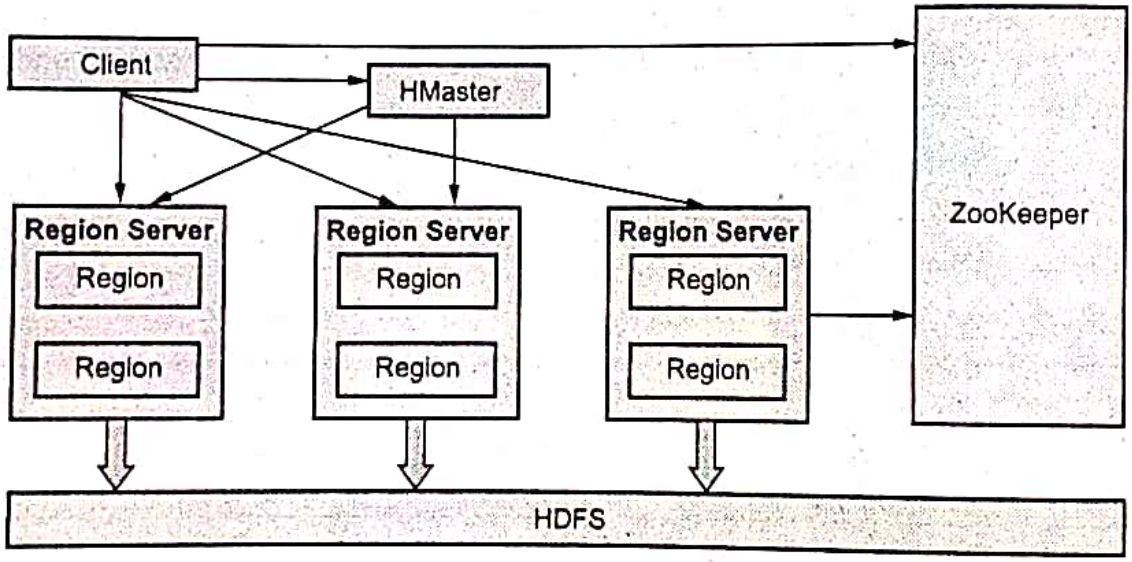


Fig. 5.1.1 Hbase architecture

- Zookeeper is a centralized monitoring server which maintains configuration information and provides distributed synchronization. If the client wants to communicate with regions servers, client has to approach Zookeeper.
- HMaster in the master server of Hbase and it coordinates the HBase cluster. HMaster is responsible for the administrative operations of the cluster.

- HRegions servers : It will perform the following functions in communication with HMaster and Zookeeper.
  1. Hosting and managing regions.
  2. Splitting regions automatically.
  3. Handling read and writes requests.
  4. Communicating with clients directly
- HRegions : For each column family, HRegions maintain a store. Main components of HRegions are Memstore and Hfile.
- Data model in HBase is designed to accommodate semi-structured data that could vary in field size, data type and columns.
- HBase is a column-oriented, non-relational database. This means that data is stored in individual columns and indexed by a unique row key. This architecture allows for rapid retrieval of individual rows and columns and efficient scans over individual columns within a table.
- Both data and requests are distributed across all servers in an HBase cluster, allowing user to query results on petabytes of data within milliseconds. HBase is most effectively used to store non-relational data, accessed via the HBase API.

5.1.1 Features and Application of Hbase

Features of Hbase :

1. Hbase is linearly scalable.
2. It has automatic failure support.
3. It provides consistent read and writes.
4. It integrates with Hadoop, both as a source and a destination.
5. It has easy java API for client.
6. It provides data replication across clusters.

Where to use Hbase ?

1. Apache Hbase is used to have random, real-time read/write access to Big Data.
2. It hosts very large tables on top of clusters of commodity hardware.
3. 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 :

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

5.1.2 Difference between HDFS and Hbase

Sr. No.	HDFS	HBase
1.	HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
2.	HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
3.	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).
4.	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.
5.	HDFS are suited for high latency operations.	HBase is suited for low latency operations.
6.	In HDFS, data are primarily accessed through Map Reduce jobs.	HBase provides access to single rows from billions of records.
7.	HDFS doesn't have the concept of random read and write operations.	HBase data is accessed through shell commands, client API in Java, REST, Avro or Thrift.

5.1.3 Difference between Hbase and Relational Database

Sr. No.	Hbase	Relational Database
1.	HBase is Schema-less	Relational Database is based on a Fixed Schema.
2.	It is Column - oriented datastore.	It is Row - oriented datastore.
3.	It is designed to store denormalized data.	It is designed to store normalized data.
4.	It contains wide and sparsely populated tables	It contains thin tables

5.	Hbase supports automatic partitioning	Relational database has no built-in support for partitioning.
6.	It is good for semi-structured as well as structured data.	It is good for structured data.
7.	No transactions are there in HBase.	RDBMS is transactional.

5.1.4 Limitations of HBase

- It takes a very long time to recover if the HMaster goes down. It takes a long time to activate another node if the first nodes go down.
- In HBase, cross data operations and join operations are very difficult to perform.
- HBase needs a new format when we want to migrate from RDBMS external sources to HBase servers.
- It is very challenging in HBase to support querying process.
- It takes enormous time to develop security factor to grant access to the users.
- HBase allows only one default sort for a table and it does not support large size of binary files.
- HBase is expensive in terms of hardware requirement and memory blocks' allocations.

5.2 Data Model and Implementations

- The Apache HBase Data Model is designed to accommodate structured or semi-structured data that could vary in field size, data type and columns. HBase stores data in tables, which have rows and columns. The table schema is very different from traditional relational database tables.
- A database consists of multiple tables. Each table consists of multiple rows, sorted by row key. Each row contains a row key and one or more column families.
- Each column family is defined when the table is created. Column families can contain multiple columns. (family : column). A cell is uniquely identified by (table,row,family : column). A cell contains an uninterpreted array of bytes and a timestamp.
- HBase data model has some logical components which are as follows :
  1. Tables
  2. Rows
  3. Column Families/Columns
  4. Versions/Timestamp
  5. Cells
- **Tables :** The HBase Tables are more like logical collection of rows stored in separate partitions called Regions. As shown above, every Region is then served by exactly one Region Server.

- The syntax to create a table in HBase shell is shown below.

```
create '<table name>','<column family>'
```

- Example : create 'CustomerContactInformation',' CustomerName' , ' ContactInfo'
- Tables are automatically partitioned horizontally by HBase into regions. Each region comprises a subset of a table's rows. A region is denoted by the table it belongs to. Fig. 5.2.1 shows region with table.

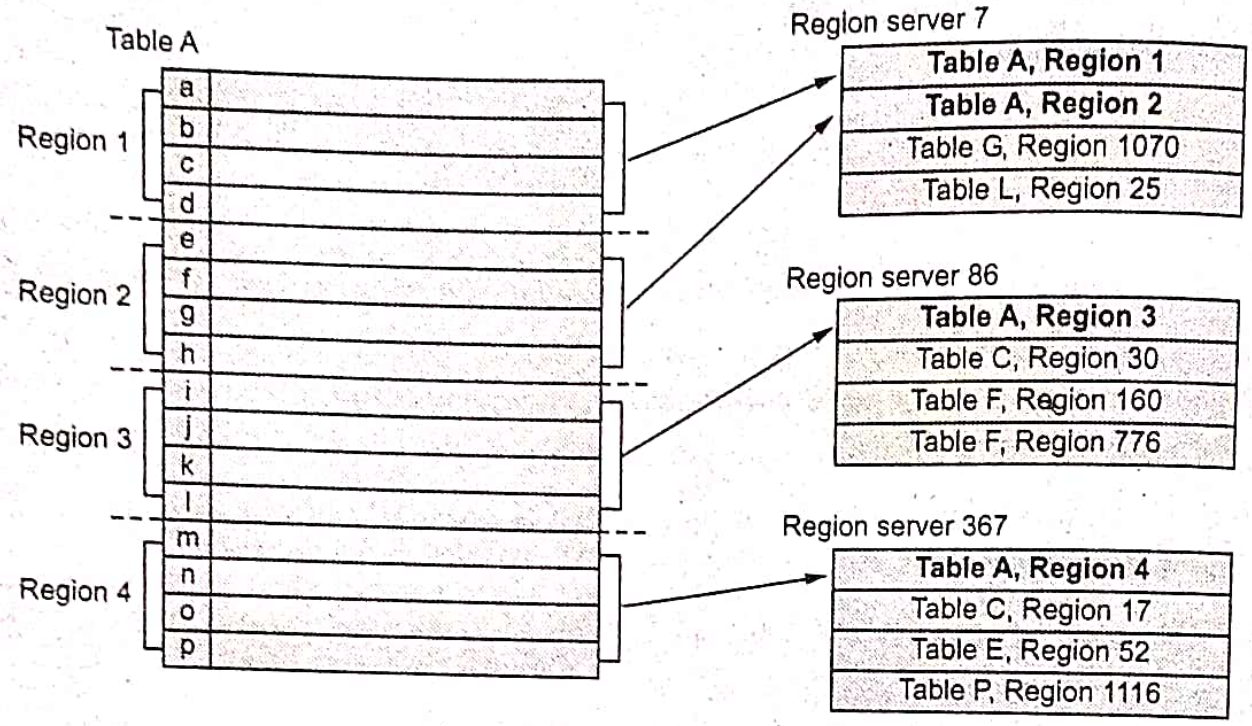


Fig. 5.2.1 Region with table

- There is one region server per node. There are many regions in a region server. At any time, a given region is pinned to a particular region server. Tables are split into regions and are scattered across region servers. A table must have at least one region.
- **Rows :** A row is one instance of data in a table and is identified by a rowkey. Rowkeys are unique in a Table and are always treated as a byte[ ].
- **Column families :** Data in a row are grouped together as Column Families. Each Column Family has one more Columns and these Columns in a family are stored together in a low level storage file known as HFile. Column Families form the basic unit of physical storage to which certain HBase features like compression are applied.
- **Columns :** A Column Family is made of one or more columns. A Column is identified by a Column Qualifier that consists of the Column Family name concatenated with the Column name using a colon - example : columnfamily : columnname. There can be multiple Columns within a Column Family and Rows within a table can have varied number of Columns.

- **Cell** : A Cell stores data and is essentially a unique combination of rowkey, Column Family and the Column (Column Qualifier). The data stored in a Cell is called its value and the data type is always treated as byte[ ].
- **Version** : The data stored in a cell is versioned and versions of data are identified by the timestamp. The number of versions of data retained in a column family is configurable and this value by default is 3.
- **Time-to-Live** : TTL is a built-in feature of HBase that ages out data based on its timestamp. This idea comes in handy in use cases where data needs to be held only for a certain duration of time. So, if on a major compaction the timestamp is older than the specified TTL in the past, the record in question doesn't get put in the HFile being generated by the major compaction; that is, the older records are removed as a part of the normal upkeep of the table.
- If TTL is not used and an aging requirement is still needed, then a much more I/O intensive operation would need to be done

5.3 Hbase Clients

- There are a number of client options for interacting with an HBase cluster. There are a number of client options for interacting with an HBase cluster.

1. Java

- Hbase is written in Java.
- Example : Creating table and inserting data in Hbase table are shown in the following program.

```
public class ExampleClient
{
    public static void main (String[] args) throws IOException
    {
        Configuration config = HBaseConfiguration.create();
        // Create table
        HBaseAdmin admin = new HBaseAdmin(config);
        HTableDescriptor htd = new HTableDescriptor("test");
    HColumnDescriptor hcd = new HColumnDescription("data");
    htd.addFamily(hcd);
    admin.createTable(htd);

    byte [] tablename = htd.getName();

    // Run some operations -- a put
    Htable table = new HTable(config, tablename);
    byte [] row1 = Bytes.toBytes("row1");
    put p1 = new put(row1);
```

```
        byte [] databytes = Bytes.toBytes("data");
        p1.add(databytes, Bytes.toBytes("FN"), Bytes.toBytes("value1"));
        table.put(p1);
        }
    }
```

- To create a table, we need to first create an instance of HBaseAdmin and then ask it to create the table named test with a single column family named data.

2. MapReduce

- HBase classes and utilities in the org.apache.hadoop.hbase.mapreduce package facilitate using HBase as a source and/or sink in MapReduce jobs. The TableInputFormat class makes splits on region boundaries so maps are handed a single region to work on. TheTableOutputFormat will write the result of MapReduce into HBase.
- Example : A MapReduce application to count the number of rows in an HBase table

```
public class RowCounter {
static final String NAME = "rowcounter";
static class RowCounterMapper
extends TableMapper<ImmutableBytesWritable, Result> {
/** Counter enumeration to count the actual rows. */
public static enum Counters {ROWS}
@Override
public void map(ImmutableBytesWritable row, Result values,
Context context)
throws IOException {
for (KeyValue value: values.list()) {
if (value.getValue().length > 0) {
context.getCounter(Counters.ROWS).increment(1);
break;
} } } }
public static Job createSubmittableJob(Configuration conf, String[] args)
throws IOException {
                String tableName = args[0];
                Job job = new Job(conf, NAME + "_" + tableName);
                job.setJarByClass(RowCounter.class);
                // Columns are space delimited
                StringBuilder sb = new StringBuilder();
                final int columnoffset = 1;
                for (int i = columnoffset; i < args.length; i++) {
                if (i > columnoffset) {
                                sb.append(" ");
                            }
                sb.append(args[i]);
            }
```

```
Scan scan = new Scan();
scan.setFilter(new FirstKeyOnlyFilter());
if (sb.length() > 0) {
for (String columnName :sb.toString().split(" ")) {
String [] fields = columnName.split(":");
if(fields.length == 1) {
scan.addFamily(Bytes.toBytes(fields[0]));
} else {
        scan.addColumn(Bytes.toBytes(fields[0]), Bytes.toBytes(fields[1]));
    } } }
// Second argument is the table name.
job.setOutputFormatClass(NullOutputFormat.class);
TableMapReduceUtil.initTableMapperJob(tableName, scan,
RowCounterMapper.class, ImmutableBytesWritable.class, Result.class, job);
job.setNumReduceTasks(0);
return job;
}
public static void main(String[] args) throws Exception {
Configuration conf = HBaseConfiguration.create();
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if (otherArgs.length < 1) {
System.err.println("ERROR: Wrong number of parameters: " + args.length);
System.err.println("Usage: RowCounter <tablename> [<column1> <column2>...]");
System.exit(-1);
}
        Job job = createSubmittableJob(conf, otherArgs);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}}
```

3. Avro, REST, and Thrift

- HBase ships with Avro, REST and Thrift interfaces. These are useful when the interacting application is written in a language other than Java. In all cases, a Java server hosts an instance of the HBase client brokering application Avro, REST, and Thrift requests in and out of the HBase cluster. This extra work proxying requests and responses means these interfaces are slower than using the Java client directly.
- REST : To put up a stargate instance, start it using the following command:

```
% hbase-daemon.sh start rest
```

- This will start a server instance, by default on port 8080, background it and catch any emissions by the server in logfiles under the HBase logs directory.
- Clients can ask for the response to be formatted as JSON, Google's protobufs, or as XML, depending on how the client HTTP Accept header is set.
- To stop the REST server, type :

```
% hbase-daemon.sh stop rest
```

- **Thrift :** Start a Thrift service by putting up a server to field Thrift clients by running the following :

```
% hbase-daemon.sh start thrift
```

- This will start the server instance, by default on port 9090, background it and catch any emissions by the server in logfiles under the HBase logs directory. The HBase Thrift documentation* notes the Thrift version used generating classes.
- To stop the Thrift server, type :

```
% hbase-daemon.sh stop thrift
```

- Avro : The Avro server is started and stopped in the same manner as we start and stop the Thrift or REST services. The Avro server by default uses port 9090.

5.4 Praxis

- When Hbase cluster running under load, following issues are considered :
  1. **Versions :** A particular Hbase version would run on any Hadoop that had a matching minor version. HBase 0.20.5 would run on an Hadoop 0.20.2, but HBase 0.19.5 would not run on Hadoop 0.20.0
  2. **HDFS :** In MapReduce, HDFS files are opened, with their content streamed through a map task and then closed. In HBase, data files are opened on cluster startup and kept open. Because of this, HBase tends to see issues not normally encountered by MapReduce clients.
- Running out of file descriptors : Because of open files on a loaded cluster, it doesn't take long before we run into system - and Hadoop - imposed limits. Each open file consumes at least one descriptor over on the remote datanode. The default limit on the number of file descriptors per process is 1,024. HBase process is running with sufficient file descriptors by looking at the first few lines of a regionservers log.
- Running out of datanode threads : The Hadoop datanode has an upper bound of 256 on the number of threads it can run at any one time.
- Sync : We must run HBase on an HDFS that has a working sync. Otherwise, there is loss of data. This means running HBase on Hadoop 0.21.x, which adds a working sync/append to Hadoop 0.20.
- UI : HBase runs a web server on the master to present a view on the state of running cluster. By default, it listens on port 60010. The master UI displays a list of basic attributes such as software versions, cluster load, request rates, lists of cluster tables and participating regionservers.

- Schema Design : HBase tables are like those in an RDBMS, except that cells are versioned, rows are sorted and columns can be added on the fly by the client as long as the column family they belong to preexists.
- Joins : There is no native database join facility in HBase, but wide tables can make it so that there is no need for database joins pulling from secondary or tertiary tables. A wide row can sometimes be made to hold all data that pertains to a particular primary key.

5.5 Pig

- Pig is an open-source high level data flow system. A high-level platform for creating MapReduce programs used in Hadoop. It translates into efficient sequences of one or more MapReduce jobs.
- Pig offers a high-level language to write data analysis programs which we call as Pig Latin. The salient property of pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.
- Pig makes use of both, the Hadoop Distributed File System as well as the MapReduce.

Features of Pig Hadoop :

1. In-built operators : Apache Pig provides a very good set of operators for performing several data operations like sort, join, filter, etc.
2. Ease of programming.
3. Automatic optimization : The tasks in Apache Pig are automatically optimized.
4. Handles all kinds of data : Apache Pig can analyze both structured and unstructured data and store the results in HDFS.

- Fig. 5.5.1 shows Pig architecture. (Refer Fig. 5.5.1 on next page)
- Pig has two execution modes :

1. **Local mode :** To run pig in local mode, we need access to a single machine; all files are installed and run using local host and file system. Specify local mode using the -x flag (pig-x local).
2. **Mapreduce mode :** To run pig in mapreduce mode, we need access to a Hadoop cluster and HDFS installation. Mapreduce mode is the default mode; but don't need to, specify it using the -x flag

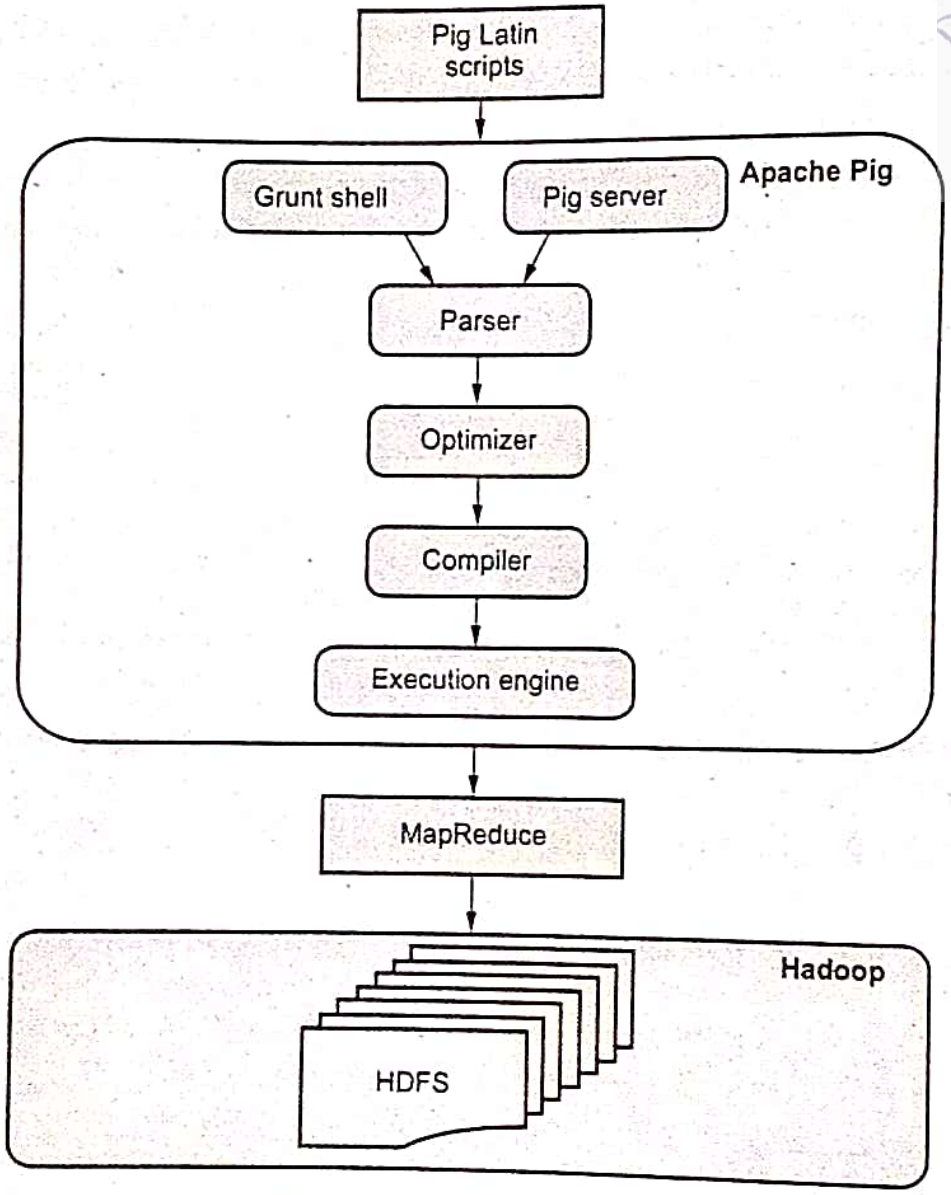


Fig. 5.5.1 Pig architecture

- Pig Hadoop framework has four main components :
  1. **Parser :** When a Pig Latin script is sent to Hadoop Pig, it is first handled by the parser. The parser is responsible for checking the syntax of the script, along with other miscellaneous checks. Parser gives an output in the form of a Directed Acyclic Graph (DAG) that contains Pig Latin statements, together with other logical operators represented as nodes.
  2. **Optimizer :** After the output from the parser is retrieved, a logical plan for DAG is passed to a logical optimizer. The optimizer is responsible for carrying out the logical optimizations.
  3. **Compiler :** The role of the compiler comes in when the output from the optimizer is received. The compiler compiles the logical plan sent by the optimize the logical plan is then converted into a series of MapReduce tasks or jobs.

4. **Execution Engine :** After the logical plan is converted to MapReduce jobs, these jobs are sent to Hadoop in a properly sorted order and these jobs are executed on Hadoop for yielding the desired result.

- Pig can run on two types of environments : The local environment in a single JVM or the distributed environment on a Hadoop cluster.
- Pig has variety of scalar data types and standard data processing options. Pig supports Map data; a map being a set of key - value pairs.
- Most pig operators take a relation as an input and give a relation as the output. It allows normal arithmetic operations and relational operations too.
- Pig's language layer currently consists of a textual language called **Pig Latin.** Pig Latin is a data flow language. This means it allows users to describe how data from one or more inputs should be read, processed and then stored to one or more outputs in parallel.
- These data flows can be simple linear flows, or complex workflows that include points where multiple inputs are joined and where data is split into multiple streams to be processed by different operators. To be mathematically precise, a Pig Latin script describes a directed acyclic graph (DAG), where the edges are data flows and the nodes are operators that process the data.
- The first step in a Pig program is to LOAD the data, which we want to manipulate from HDFS. Then run the data through a set of transformations. Finally, DUMP the data to the screen or STORE the results in a file somewhere.

Advantages of Pig :

1. Fast execution that works with MapReduce, Spark and Tez.
2. Its ability to process almost any amount of data, regardless of size.
3. A strong documentation process that helps new users learn Pig Latin.
4. Local and remote interoperability that lets professionals work from anywhere with a reliable connection.

Pig disadvantages :

1. Slow start-up and clean-up of MapReduce jobs
2. Not suitable for interactive OLAP analytics
3. Complex applications may require many user defined function.

5.5.1 Pig Data Model

- With Pig, when the data is loaded the data model is specified. Any data that we load from the disk into Pig will have a specific schema and structure. Pig data model is rich enough to manage most of what's thrown in its way like table - like structures and nested hierarchical data structures.
- However, Pig data types can be divided into two groups in general terms : Scalar forms and complex types.
- Scalar types contain a single value, while complex types include other values, such as the values of Tuple, Container and Map.
- In its data model, Pig Latin has those four types :
  - Atom : An atom is any single attribute, like, for example, a string or a number 'Hadoop'. The atomic values of Pig are scalar forms that appear, for example, in most programming languages, int, long, float, double, char array and byte array.
  - Tuple : A tuple is a record generated by a series of fields. For example, each field can be of any form, 'Hadoop,' ,' or 6. Just think of a tuple in a table as a row.
  - Bag : A pocket is a set of tuples, which are not special. The bag's schema is flexible, each tuple in the set can contain an arbitrary number of fields and can be of any sort.
  - Map : A map is a set of pairs with main values. The value can store any type and the key needs to be unique. A char array must be the key of a map and the value may be of any kind.

5.5.2 Pig Latin

- The Pig Latin is a data flow language used by Apache Pig to analyze the data in Hadoop. It is a textual language that abstracts the programming from the Java MapReduce idiom into a notation.
- The Pig Latin statements are used to process the data. It is an operator that accepts a relation as an input and generates another relation as an output.
  a) It can span multiple lines.
  b) Each statement must end with a semi-colon.
  c) It may include expression and schemas.
  d) By default, these statements are processed using multi - query execution
- Pig Latin statements work with relations. A relation can be defined as follows :
  a) A relation is a bag (more specifically, an outer bag).
  b) A bag is a collection of tuples.

c) A tuple is an ordered set of fields.

d) A field is a piece of data.

- **Pig Latin Datatypes**

1. Int : "Int" represents a signed 32-bit integer. For Example : 13
2. Long : It represents a signed 64-bit integer. For Example : 13L
3. Float : This data type represents a signed 32-bit floating point. For Example : 130.5F
4. Double : "double" represents a 64-bit floating point. For Example : 13.5
5. Chararray : It represents a character array (string) in Unicode UTF-8 format. For Example : 'Big Data'
6. Bytearray : This data type represents a Byte array.
7. Boolean : "Boolean" represents a Boolean value. For Example : true/ false.

5.5.3 Developing and Testing Pig Latin Scripts

- Pig provides several tools and diagnostic operators to help us to develop applications.
- Scripts in Pig can be executed in interactive or batch mode. To use pig in interactive mode, we invoke it in local or map-reduce mode then enter commands one after the other. In batch mode, we save commands in a .pig file and specify the path to the file when invoking pig.
- At an overly simplified level a Pig script consists of three steps. In the first step we load data from HDFS. In the second step we perform transformations on the data. In the final step we store transformed data. Transformations are the heart of Pig scripts.
- Pig has a schema concept that is used when loading data to specify what it should expect. First specify columns and optionally their data types. Any columns in data but not included in the schema are truncated.
- When we have fewer columns than those specified in schema they are filled with nulls. To load sample data sets we first move them to HDFS then from there we will load into Pig.
- **Pig Script Interfaces :** Pig programs can be packaged in three different ways.

1. Script : This function is nothing more than a file consists of Pig Latin commands, identified by the **.pig** suffix. Ending Pig program with the .pig extension is a convention but not required. The commands are interpreted by the Pig Latin compiler and then runs in the order determined by the Pig optimizer.

2. Grunt : Grunt acts as a command interpreter where we can interactively enter Pig Latin at the Grunt command line and immediately see the response. This method is useful for prototyping during early development stage and with what-if scenarios.
3. Embedded : Pig Latin statements can run within Java, JavaScript and Python programs.

- Pig scripts, Grunt shell Pig commands and embedded Pig programs can be executed in either Local mode or on MapReduce mode. The Grunt shell enables an interactive shell to submit Pig commands and run Pig scripts. To start the Grunt shell in Interactive mode, we need to submit the command pig at the shell.
- To tell the complier whether a script or Grunt shell is executed locally or in Hadoop mode just specify it in the -x flag to the pig command. The following is an example of how we would specify running our Pig script in local mode :

```
pig -x local mindStick.pig
```

- Here's how we would run the Pig script in Hadoop mode, which is the default if we don't specify the flag :

```
pig -x mapreduce mindstick.pig
```

- By default, when we specify the pig command without any parameters, it starts the Grunt shell in Hadoop mode. If we want to start the Grunt shell in local mode just add the -x local flag to the command.

5.6 Hive

- Apache Hive is an open source data warehouse software for reading, writing and managing large data set files that are stored directly in either the Apache Hadoop Distributed File System (HDFS) or other data storage systems such as Apache HBase.
- Data analysts often use Hive to analyze data, query large amounts of unstructured data and generate data summaries.
- Features of Hive :
  1. It stores schema in a database and processes data into HDFS.
  2. It is designed for OLAP.
  3. It provides SQL type language for querying called HiveQL or HQL.
  4. It is familiar, fast, scalable and extensible.
- Hive supports variety of storage formats : TEXTFILE for plaintext, SEQUENCEFILE for binary key-value pairs, RCFILE stores columns of a table in a record columnar format

- Hive table structure consists of rows and columns. The rows typically correspond to some record, transaction, or particular entity detail.
- The values of the corresponding columns represent the various attributes or characteristics for each row.
- Hadoop and its ecosystem are used to apply some structure to unstructured data. Therefore, if a table structure is an appropriate way to view the restructured data, Hive may be a good tool to use.
- Following are some Hive use cases :
  1. Exploratory or ad-hoc analysis of HDFS data : Data can be queried, transformed and exported to analytical tools.
  2. Extracts or data feeds to reporting systems, dashboards, or data repositories such as HBase.
  3. Combining external structured data to data already residing in HDFS.

Advantages :

1. Simple querying for anyone already familiar with SQL.
2. Its ability to connect with a variety of relational databases, including Postgres and MySQL.
3. Simplifies working with large amounts of data.

Disadvantages :

1. Updating data is complicated
2. No real time access to data
3. High latency.

- **Program Example :** Write a code in JAVA for a simple Word Count application that counts the number of occurrences of each word in a given input set using the Hadoop Map-Reduce framework on local-standalone set-up.

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

```
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {
 public static class TokenizerMapper
     extends Mapper<Object, Text, Text, IntWritable>{
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();
  public void map(Object key, Text value, Context context )
     throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
     word.set(itr.nextToken());
     context.write(word, one);
    }
  }
 }
 public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();
  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
     sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
 }
 public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "word count");
  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

5.6.1 Hive Architecture

- Fig. 5.6.1 shows Hive architecture.

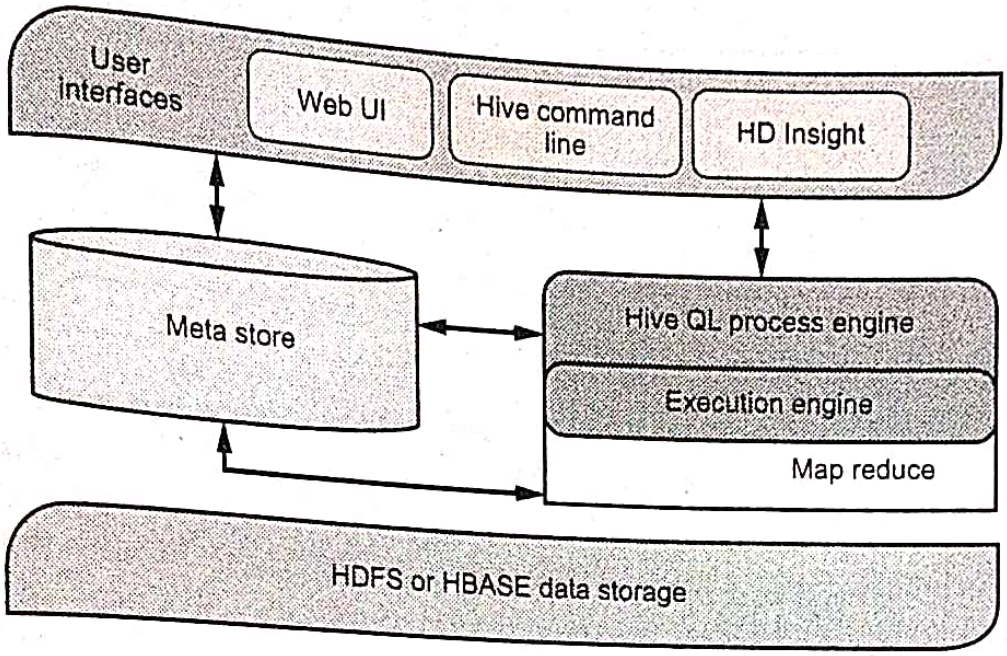


Fig. 5.6.1 Hive architecture

- User Interface : Hive is a data warehouse infrastructure software that can create interaction between user and HDFS.
- The user interfaces that Hive supports are Hive Web UI, Hive command line and Hive HD Insight.
- Meta Store : Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types and HDFS mapping.
- HiveQL Process Engine : HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
- Execution engine : The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
- HDFS or HBASE : Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Working of Hive :

- Fig. 5.6.2 shows Hive working.

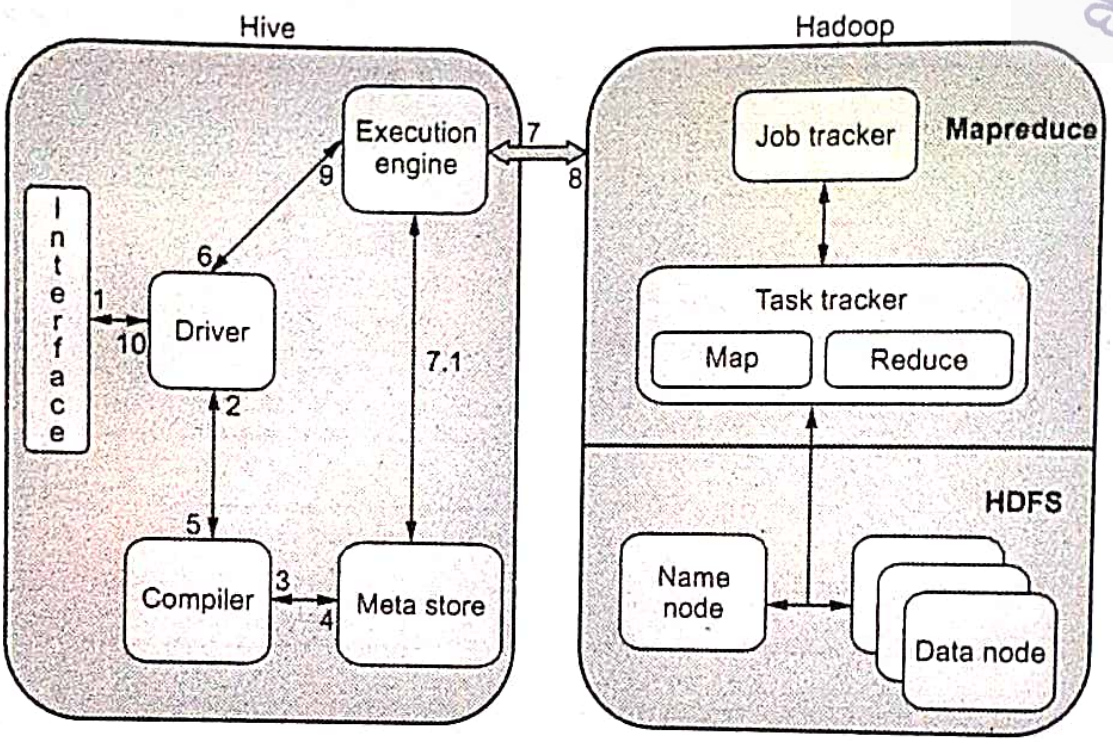


Fig. 5.6.2 Hive working

1. Execute query : The Hive interface such as command line or Web UI sends query to driver to execute.
2. Get plan : The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3. Get metadata : The compiler sends metadata request to metastore.
4. Send metadata : Metastore sends metadata as a response to the compiler.
5. Send plan : The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6. Execute plan : The driver sends the execute plan to the execution engine.
7. Execute job : Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.

   7.1 Metadata Ops : Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8. Fetch result : The execution engine receives the results from data nodes.
9. Send results : The execution engine sends those resultant values to the driver.
10. Send results : The driver sends the results to Hive Interfaces

5.6.2 Data Types and File Formats

1. Data types :

- Hive data types can be classified into two categories : Primary data types and Complex data types.
- **Primary data types are of four types :** Numeric, string, date/time and miscellaneous types
- Numeric data types : Integral types are TINYINT, SMALLINT, INT and BIGINT. Floating types are FLOAT, DOUBLE and DECIMAL.
- **String data types are** string, varchar and char.
- **Date/Time data types :** Hive provides DATE and TIMESTAMP data types in traditional UNIX time stamp format for date/time related fields in hive. DATE values are represented in the form YYYY--MM--DD. TIMESTAMP use the format yyyy-mm-dd hh:mm:ss[.f...].
- **Miscellaneous types :** Hive supports two more primitive data types : BOOLEAN and BINARY. Hive stores true or false values only.

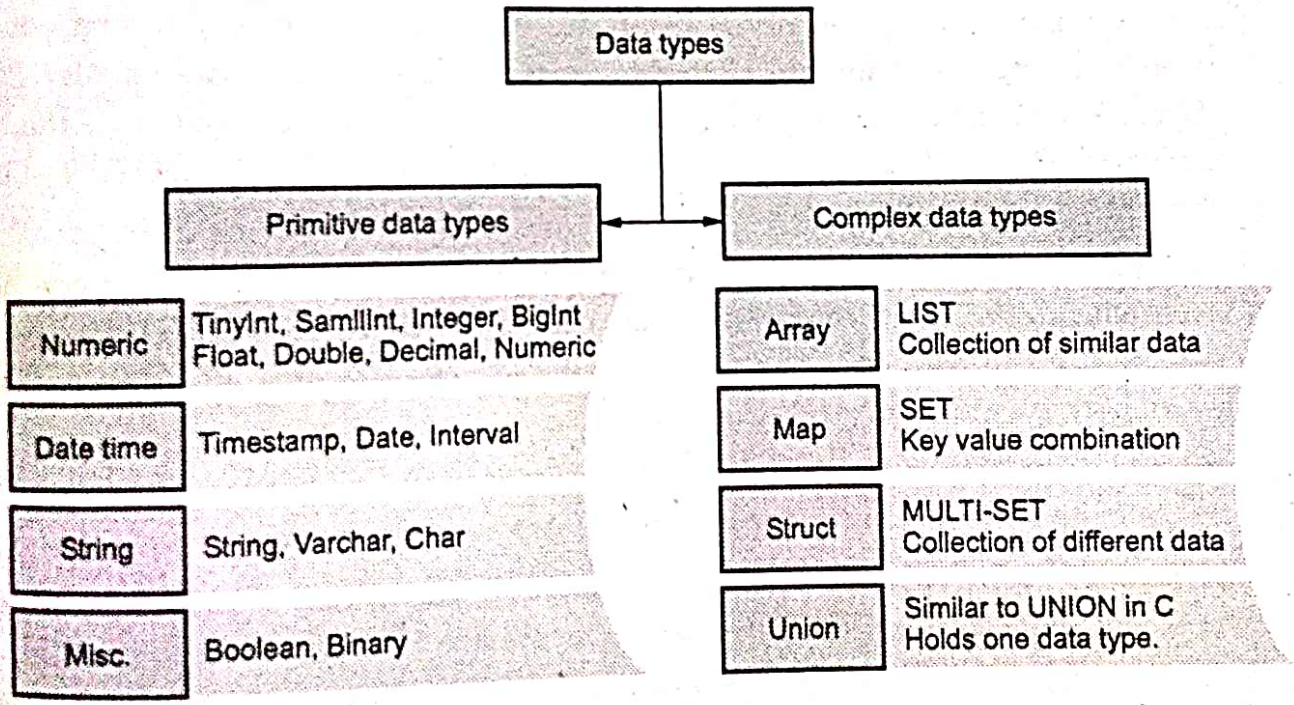


Fig. 5.6.3

- **Complex type are** Array, Map, Struct and Union.
  - Array in Hive is an ordered sequence of similar type elements that are indexable using the zero-based integers.
  - Map in Hive is a collection of key-value pairs, where the fields are accessed using array notations of keys (e.g., ['key']).

- STRUCT in Hive is similar to the STRUCT in C language. It is a record type that encapsulates a set of named fields, which can be any primitive data type.
- UNION type in Hive is similar to the UNION in C. UNION types at any point of time can hold exactly one data type from its specified data types.

2. File formats :

- In Hive it refers to how records are stored inside the file. As we are dealing with structured data, each record has to be its own structure. How records are encoded in a file defines a file format. These file formats mainly vary between data encoding, compression rate, usage of space and disk I/O.
- Hive support file format : TEXTFILE, SEQUENCEFILE, RCFILE and ORCFILE.
- TEXTFILE format is a famous input/output format used in Hadoop. In Hive if we define a table as TEXTFILE it can load data of from CSV (Comma Separated Values), delimited by Tabs, Spaces and JSON data.
- Sequence files are flat files consisting of binary key - value pairs. When Hive converts queries to MapReduce jobs, it decides on the appropriate key - value pairs to be used for a given record. Sequence files are in the binary format which can be split and the main use of these files is to club two or more smaller files and make them as a one sequence file. In Hive we can create a sequence file by specifying STORED AS SEQUENCEFILE in the end of a CREATE TABLE statement.
- RCFILE stands of Record Columnar File which is another type of binary file format which offers high compression rate on the top of the rows. RCFILE is used when we want to perform operations on multiple rows at a time. RCFILEs are flat files consisting of binary key/value pairs.
- Facebook uses RCFILE as its default file format for storing of data in their data warehouse as they perform different types of analytics using Hive.
- ORCFILE : ORC stands for Optimized Row Columnar which means it can store data in an optimized way than the other file formats. ORC reduces the size of the original data up to 75 %. An ORC file contains rows data in groups called as Stripes along with a file footer. ORC format improves the performance when Hive is processing the data.

5.7 HiveQL Data Definition

- HiveQL is the Hive query language. Hive offers no support for row level inserts, updates and deletes. Hive doesn't support transactions. DDL statements are used to define or change Hive databases and database objects.

- Types of Hive DDL commands are : CREATE, SHOW, DESCRIBE, USE, DROP, ALTER and TRUNCATE.
- **Hive DDL commands**

DDL command	Use with
CREATE	Database, table
SHOW	Databases, tables, Table properties, Partitions, Functions, Index
DESCRIBE	Database, Table, view
USE	Database
DROP	Database, Table
ALTER	Database, Table
TRUNCATE	Table

- Hive database : In Hive, the database is considered as a catalog or namespace of tables. It is also common to use databases to organize production tables into logical groups. If we do not specify a database, the default database is used.
- Let's create a new database by using the following command :

```
hive> CREATE DATABASE Rollcall;
```

- Make sure the database we are creating doesn't exist on Hive warehouse, if exists it throws Database Rollcall already exists error.
- At any time, we can see the databases that already exist as follows :

```
hive> SHOW DATABASES;
default
Rollcall
hive> CREATE DATABASE student;
hive> SHOW DATABASES;
default
Rollcall
student
```

- Hive will create a directory for each database. Tables in that database will be stored in subdirectories of the database directory. The exception is tables in the default database, which doesn't have its own directory.
- **Drop Database Statement :**

```
Syntax: DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS]
database_name [RESTRICT | CASCADE];
```

Example : hive> DROP DATABASE IF EXISTS userid;

- **ALTER DATABASE :** The ALTER DATABASE statement in Hive is used to change the metadata associated with the database in Hive. Syntax for changing Database Properties :

```
ALTER (DATABASE|SCHEMA) database_name SET DBPROPERTIES
(property_name=property_value, ...);
```

5.8 HiveQL Data Manipulation

- Data manipulation language is a subset of SQL statements that modify the data stored in tables. Hive has no row - level insert, update and delete operations, the only way to put data into an table is to use one of the "bulk" load operations.

Inserting data into tables from queries :

- The INSERT statement perform loading data into a table from a query.

```
INSERT OVERWRITE TABLE students
PARTITION (branch = 'CSE', classe = 'OR')
SELECT * FROM college_students se
WHERE se.bra = 'CSE' AND se.cla = 'OR';
```

- With OVERWRITE, any previous contents of the partition are replaced. If we drop the keyword OVERWRITE or replace it with INTO, Hive appends the data rather than replaces it. This feature is only available in Hive v0.8.0 or later.
- We can mix INSERT OVERWRITE clauses and INSERT INTO clauses, as well.

Dynamic partition inserts :

- Hive also supports a dynamic partition feature, where it can infer the partitions to create based on query parameters. Hive determines the values of the partition keys, from the last two columns in the SELECT clause.
- The static partition keys must come before the dynamic partition keys. Dynamic partitioning is not enabled by default. When it is enabled, it works in "strict" mode by default, where it expects at least some columns to be static. This helps protect against a badly designed query that generates a gigantic number of partitions.
- **Hive Data Manipulation Language (DML) Commands**
  a) LOAD - The LOAD statement transfers data files into the locations that correspond to Hive tables.
  b) SELECT - The SELECT statement in Hive functions similarly to the SELECT statement in SQL. It is primarily for retrieving data from the database.
  c) INSERT - The INSERT clause loads the data into a Hive table. Users can also perform an insert to both the Hive table and/or partition.

d) DELETE - The DELETE clause deletes all the data in the table. Specific data can be targeted and deleted if the WHERE clause is specified.

e) UPDATE - The UPDATE command in Hive updates the data in the table. If the query includes the WHERE clause, then it updates the column of the rows that meet the condition in the WHERE clause.

f) EXPORT - The Hive EXPORT command moves the table or partition data together with the metadata to a designated output location in the HDFS.

g) IMPORT - The Hive IMPORT statement imports the data from a particularized location to a new or currently existing table.

5.9 HiveQL Queries

- The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore. Hive Query Language is used for processing and analyzing structured data. It separates users from the complexity of Map Reduce programming.

SELECT ... FROM Clauses

- SELECT is the projection operator in SQL. The FROM clause identifies from which table, view or nested query we select records. For a given record, SELECT specifies the columns to keep, as well as the outputs of function calls on one or more columns.
- Here's the syntax of Hive's SELECT statement.

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[CLUSTER BY col_list | [DISTRIBUTE BY col_list] [SORT BY col_list]]
[LIMIT number]
;
```

- SELECT is the projection operator in HiveQL. The points are :
  a) SELECT scans the table specified by the FROM clause
  b) WHERE gives the condition of what to filter
  c) GROUP BY gives a list of columns which specify how to aggregate the records
  d) CLUSTER BY, DISTRIBUTE BY, SORT BY specify the sort order and algorithm
  e) LIMIT specifies how many # of records to retrieve.

Computing with Columns

- When we select the columns, we can manipulate column values using either arithmetic operators or function calls. Math, date and string functions are also popular.
- Here's an example query that uses both operators and functions.

```
SELECT upper(name), sales_cost FROM products;
```

- WHERE Clauses : A WHERE clause is used to filter the result set by using predicate operators and logical operators. Functions can also be used to compute the condition.
- GROUP BY Clauses : A GROUP BY clause is frequently used with aggregate functions, to group the result set by columns and apply aggregate functions over each group. Functions can also be used to compute the grouping key.

5.10 Two Marks Questions with Answers

Q.1 What is HBase ?

Ans. : 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.

Q.2 What is Hive ?

Ans. : Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries and the analysis of large datasets stored in Hadoop compatible file systems. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL

Q.3 What is Hive data definition ?

Ans. : Hive data definition assigns relational structure to the files stored on the HDFS cluster. We can easily query the structured data to extract specific information. For example, data definition for log files would contain columns like : CLASS, FILENAME, MESSAGE, LINENUBER, etc.

Q.4 Explain services provided by Zookeeper in Hbase.

Ans. : Various services that Zookeeper provides include :

a) Establishing client communication with region servers.

b) Tracking server failure and network partitions.

c) Maintain configuration information

d) Provides ephemeral nodes, which represent different region servers.

Q.5 What is Zookeeper ?

Ans. : ZooKeeper service keeps track of all the region servers that are there in an HBase cluster - tracking information about how many region servers are there and which region servers are holding which DataNode.

Q.6 What are the responsibilities of HMaster ?

Ans. : Responsibilities of HMaster :

a) Manages and monitors the Hadoop cluster

b) Performs administration

c) Controlling the failover

d) DDL operations are handled by the HMaster

e) Whenever a client wants to change the schema and change any of the metadata operations, HMaster is responsible for all these operations.

Q.7 Where to Use HBase ?

Ans. : Hadoop HBase is used to have random real - time access to the big data. It can host large tables on top of cluster commodity. HBase is a non - relational database which modelled after Google's big table. It works similar to a big table to store the files of Hadoop.

Q.8 Explain unique features of Hbase ?

Ans.:

- HBase is built for low latency operations
- HBase is used extensively for random read and write operations
- HBase stores a large amount of data in terms of tables
- Automatic and configurable sharding of tables
- HBase stores data in the form of key/value pairs in a columnar model

Q.9 Explain data model in Hbase ?

Ans. : The data model in HBase is designed to accommodate semi - structured data that could vary in field size, data type and columns. Additionally, the layout of the data model makes it easier to partition the data and distribute it across the cluster.

Q.10 What is the difference between Pig Latin and Pig engine ?

Ans. : Pig Latin is a scripting language similar to Perl used to search large data sets. It is composed of a sequence of transformations and operations that are applied to the input data to create data.

The Pig engine is the environment in which Pig Latin programs are executed. It translates Pig Latin operators into MapReduce jobs.

Q.11 What is pig storage ?

Ans. : Pig has a built-in load function called pig storage. In addition, whenever we wish to import data from a file system into the Pig, we can use Pig storage.

Q.12 What are the features of Hive ?

Ans. :

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable and extensible.

❑❑❑

SOLVED MODEL QUESTION PAPER

[As Per New Syllabus]

Big Data Analytics

Semester - V (AI&DS)
Vertical - 1 (Verticals for AIDS I) (AI&DS)
Vertical - 1 (Data Science) (CSE/IT/CS&BS)
Vertical - VI (Diversified Courses) (EEE)

Time : Three Hours] [Maximum Marks : 100

Answer ALL Questions

PART A - (10 × 2 = 20 Marks)

Q.1 *What is Hadoop ?* **(Refer Two Marks Q.15 of Chapter - 1)**

Q.2 *What is data science ?* **(Refer Two Marks Q.1 of Chapter - 1)**

Q.3 *Explain Cassandra data center.* **(Refer Two Marks Q.10 of Chapter - 2)**

Q.4 *What is the difference between Sharding and replication ?* **(Refer Two Marks Q.4 of Chapter - 2)**

Q.5 *Why is a block in HDFS so large ?* **(Refer Two Marks Q.5 of Chapter - 3)**

Q.6 *What is MapFile ?* **(Refer Two Marks Q.11 of Chapter - 3)**

Q.7 *Define MapReduce.* **(Refer Two Marks Q.1 of Chapter - 4)**

Q.8 *Explain First In First Out (FIFO) scheduling.* **(Refer Two Marks Q.7 of Chapter - 4)**

Q.9 *What is pig storage ?* **(Refer Two Marks Q.11 of Chapter - 5)**

Q.10 *What is Zookeeper ?* **(Refer Two Marks Q.5 of Chapter - 5)**

PART B - (5 × 13 = 65 Marks)

Q.11 a) i) *What is unstructured data? Compare structured and unstructured data.* **(Refer section 1.3)** [6]

ii) *Explain application of big data.* **(Refer section 1.6)** [7]

OR

b) i) *What is web analytics? Why web analytics is important ?* **(Refer section 1.5)** [6]

ii) *Draw and explan Hadoop ecosystem.* **(Refer section 1.8.1)** [7]

Q.12 a) i) *Briefly discuss schemaless database.* **(Refer section 2.3)** [7]

ii) *What is CAP theorem? Explain.* **(Refer section 2.1.3)** [6]

OR

b) i) *What Sharding? Compare Sharding with replication.* **(Refer sections 2.5.2 nad 2.5.6)** [6]

ii) *Discuss read and write Quorums.* **(Refer section 2.6.3)** [7]

Q.13 a) i) *What is Hadoop streaming? Explain feature of Hadoop streaming.* **(Refer section 3.2)** [6]

ii) *Explain heartbeat mechanism of HDFS.* **(Refer section 3.4.5)** [7]

OR

b) *Explain the following (Any Three) :*

i) *Writable interface of Hadoop* **(Refer section 3.5.5)**

ii) *Avro* **(Refer section 3.5.6)**

iii) *Data integrity in HDFS* **(Refer section 3.5.1)**

iv) *Hadoop local file system* **(Refer section 3.5.2)** [13]

Q.14 a) i) *Discuss data flow in MapReduce programming model.* **(Refer section 4.1.2)** [6]

ii) *Write short note on YARN.* **(Refer section 4.4)** [7]

OR

b) i) *Discuss Input - Output format of MapReduce.* **(Refer section 4.9.1)** [6]

ii) *What is capacity scheduler? Compare capacity and fair scheduler.* **(Refer sections 4.6.3 and 4.6.4)** [7]

Q.15 a) i) *What is Hbase? Draw architecture of Hbase. Explain difference between HDFS and Hbase.* **(Refer section 5.1)** [13]

OR

b) i) *Write short note on Hbase client.* **(Refer section 5.3)** [6]

ii) *What is pig? Explain feature of pig. Draw architecture of pig.* **(Refer section 5.5)** [7]

PART C - (1 × 15 = 15 Marks)

Q.16 a) i) *What is open source technology? Explain advantages, disadvantages and application of open source.* **(Refer section 1.9)** [7]

ii) *Explain failures in classic map reduce and YARN.* **(Refer section 4.5)** [8]

OR

b) *Explain with diagram various aggregate data model of NoSQL.* **(Refer section 2.2)** [15]

❑❑❑