Part 1

Write a short report (1,200 words, ± 10%) discussing and describing the different layers and components of the big data technologies stack.

Big Data Technologies Stack

Introduction

Big data technologies have emerged as critical enablers for organizations to capture, store, process, and analyze vast amounts of data to gain valuable insights and make data-driven decisions. They consist of various layers and components that work together to handle the challenges associated with big data, such as volume, velocity, variety, and veracity. This report will provide an overview of the different layers and components, ranging from redundant physical infrastructure to big data analytics and applications. We will also highlight some popular tools available in the Apache Hadoop ecosystem that are commonly used in each layer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Interface/Feeds from/to the Internal apps |  | Big Data Applications |  | Interface/Feeds from/to the Internal apps |
| Analytics, Reporting, and Visualization |
| Analytical Data Warehouses and Data Marts |
| Organising Databases and Tools |
| Operational Databases |
| Security Infrastructure  Redundant Physical Infrastructure |

Level 0: The redundant physical infrastructure layer is the lowest level of the stack and an essential part of the big data technology stack as it ensures high availability and fault tolerance of data and processing capabilities. It is fully based on the computing model, which means data may be stored in different physical locations and can be linked together through networks. If the organization creates a private cloud, we will want to have redundancy built within the private environment so that it can get out of a position to support changing workloads. Big data environments require high performance, availability, scalability, and flexibility. Performance is essential because huge amounts of data must be processed and analysed in real-time or almost real-times. Availability is achieved by implementing redundancy and fault tolerance techniques, such as redundant servers, storage units, and network links. Scalability is achieved by scaling infrastructure both horizontally and vertically. Flexibility is essential in big data environments as data sources and processing requirements change rapidly. Technologies such as virtualization, containerization, and cloud computing offer flexibility in deploying and managing big data infrastructure.

Tools like The Hadoop Distributed File System (HDFS) provides redundant physical infrastructure capabilities. HDFS is a distributed file system that replicates data across multiple nodes in a Hadoop cluster, ensuring fault tolerance and high availability. Other tools, such as Apache ZooKeeper and Apache Ambari, also provide capabilities for managing redundant physical infrastructure in Hadoop clusters.

Level 1: Security infrastructure is a critical aspect of big data technologies, as large volumes of sensitive data can require protection from unauthorized access and breaches. This layer includes tools and technologies for authentication, authorization, encryption, and data auditing in big data environments. Common security and privacy considerations include data access, application access, data encryption, threat detection, scalability, compliance, data lifecycle management, employee access, security monitoring and auditing, and patch management.

Popular tools like Apache Ranger is a centralised security management framework that provides fine-grained authorization and access control for Hadoop and related components, while Apache Knox is a gateway that provides secure access to Hadoop clusters and other big data services through authentication and authorization mechanisms. Apache Sentry is a granular, role-based authorization framework for Hadoop that provides access control at various levels.

Vertical Strands: Interfaces and Feeds: Vertical strands refer to the data ingestion and exchange processes in big data applications, which involve interfacing with internal applications or systems. This includes ingesting data from various sources like databases, data streams, APIs, or external applications and exchanging data with other internal applications.

Bidirectional access between components in a big data stack is typically facilitated through APIs (Application Programming Interfaces). The most common architecture for APIs that connect web resources is Representational State Transfer (REST). RESTful APIs allow for efficient and scalable communication between components in a distributed big data environment, enabling data to be retrieved, processed, and shared seamlessly.

Level 2: Operational Databases: Operational databases manage real-time transactional data in big data applications. These databases are optimized for fast data ingestion, updates, and retrieval for real-time processing. The core of a big data environment is where data elements are stored, typically using a database language like Structured Query Language (SQL).

Apache Cassandra is a highly scalable and distributed NoSQL database designed for handling large volumes of data across multiple servers. Apache Hudi provides incremental data processing and data management capabilities for real-time updates. Apache Phoenix is a SQL query engine for Hadoop that enables real-time SQL-like querying on top of HBase.

Level 3: Organizing Databases and Tools in Big Data Environments involves capturing, validating, and assembling data elements into relevant collections using technologies like MapReduce in the Hadoop ecosystem. Various technologies, such as Distributed File Systems (HDFS) and Apache HBase. Serialization services are used to convert data objects into a format that can be stored, transmitted, or processed. Coordination services are necessary for managing distributed systems and ensuring synchronization and consistency among distributed components. ETL tools are essential for data extraction, transformation, and loading tasks in big data applications. Workflow services are required for defining, managing, and executing complex data processing workflows in big data applications. Popular tools include Apache HBase, Sqoop, and Atlas.

Apache HBase is a distributed NoSQL database that provides real-time, columnar, and scalable storage for large volumes of data. Apache Sqoop is a tool for data integration between Hadoop and relational databases, enabling efficient data transfer and processing. Finally, Apache Atlas is a metadata management and governance framework for Hadoop, facilitating the organization and management of data lineage, metadata, and data quality across the big data ecosystem.

Level 4: Analytical Data Warehouses and Data Marts are used for storing and managing large volumes of data for analytics and reporting purposes. In the Apache Hadoop ecosystem, popular tools for analytical data warehousing and data marts include Apache Hive, Apache Impala, and Amazon Redshift. Apache Hive is a data warehouse infrastructure built on top of Hadoop, providing SQL-like querying capabilities for large-scale data. Impala is an open-source SQL query engine that provides fast and interactive analytics for data stored in Hadoop clusters. Amazon Redshift is a cloud-based data warehousing service provided by Amazon Web Services (AWS), designed for big data analytics.

These tools provide powerful capabilities for storing, managing, and analyzing large volumes of data in analytical data warehouses and data marts within the Apache Hadoop ecosystem. They are widely used in big data applications for enabling efficient data analysis and reporting to support decision-making and business insights.

Level 5: Analytics, Reporting, and Visualisation: This is the topmost layer in the big data technology stack, where data is processed and analyzed to gain insights and generate reports. This layer includes tools and technologies for advanced analytics, data visualization, and reporting. Popular tools from the Apache Hadoop ecosystem for analytics, reporting, and visualization include Apache Zeppelin, a web-based notebook providing an interactive and collaborative environment for data analysis and visualization. Zeppelin supports multiple programming languages, such as Scala, Python, and R, and provides built-in visualizations for generating interactive charts, graphs, and dashboards.

Apache Superset is a modern and enterprise-ready data exploration and visualization platform that provides a wide range of visualization options and features.

Apache Flink is a stream processing framework that provides event time processing, state management, and windowing capabilities for real-time analytics.

Level 6: Big Data Applications: There are two types: horizontal and vertical. Horizontal applications are designed to solve problems across industries and can be used in various domains, such as customer relationship management (CRM) systems, fraud detection systems, recommendation engines, and sentiment analysis tools. Vertical applications are industry-specific and tailored to specific industries like electronic health record (EHR) systems for healthcare, supply chain management systems for logistics, predictive maintenance systems for manufacturing, and energy management systems for utilities. Big data applications can be both custom-built and third-party applications. They enable organizations to harness the power of big data and derive valuable insights from massive data sets.

In conclusion, Understanding the different layers and components of the big data technologies stack is essential for organizations looking to implement big data solutions. Organizations can effectively manage, process, analyze, and visualize big data to gain valuable insights and make data-driven decisions by leveraging the right tools and frameworks for each layer.

Part 2: a)

Briefly summarise (600 words, ± 10%) the differences between relational and non-relational (NoSQL) databases. Discuss the different use cases for both types of database, and how they are both applicable in a big data environment

Relational databases and non-relational (NoSQL) databases are two distinct types of databases that differ in their data modeling approach, data consistency, scalability, and performance characteristics.

Relational databases follow a fixed schema with tables and relationships between tables, while NoSQL databases use flexible schemas for storing data in various formats like key-value pairs, column families, documents, and graphs. One major difference is in data modeling, where relational databases have predefined schemas for data consistency and integrity. In contrast, NoSQL databases allow for flexible schemas to handle diverse and dynamic data types, such as social media data, sensor data, and multimedia content.

The use cases for relational databases are typically suited for applications that require transactional consistency and complex queries, such as online transactional processing (OLTP) systems. Relational databases excel in handling relationships between data entities and support powerful querying capabilities with SQL (Structured Query Language). As a result, they are widely used in industries such as finance, e-commerce, and human resources management, where data integrity and consistency are critical.

On the other hand, NoSQL databases are often used in applications that require high scalability, high availability, and quick data retrieval, such as online analytical processing (OLAP) systems. NoSQL databases may use different query languages, APIs, or even full-text searches for data retrieval and manipulation. NoSQL databases are particularly suitable for handling large volumes of unstructured or semi-structured data, such as in social media analytics, log file analysis, and recommendation engines. They follow the BASE (basically available, soft state, eventually consistent) properties, which prioritize availability and partition tolerance over immediate consistency. NoSQL databases are commonly used in big data environments, where scalability and performance are essential for processing large volumes of data. For example, in a social media analytics application, NoSQL databases can store data in the form of key-value pairs or documents, making it easy to store and analyze large amounts of data, including text, images, and multimedia content.

The choice between relational and NoSQL databases depends on the specific requirements of the application. Relational databases are suitable for applications that require transactional consistency and data integrity. They are commonly used in financial applications, e-commerce, and other data-intensive applications that require complex querying and data relationships. NoSQL databases, on the other hand, are suitable for applications that require scalability, high performance, and flexibility in data modeling. They are commonly used in web applications, social media platforms, IoT, and big data applications.

In a big data environment, both relational and NoSQL databases have their applicability depending on the nature of the data and the requirements of the application. Relational databases are often used for traditional data processing tasks that require consistency and complex data relationships. They are also suitable for handling structured data that follows a predefined schema. However, relational databases may struggle to handle the volume and velocity of big data and may require expensive hardware and software upgrades to scale effectively.

Relational databases are typically vertically scalable, meaning they require more powerful hardware to handle increasing workloads, while NoSQL databases provide horizontal scalability, enabling data to be distributed across a cluster of nodes, which makes them well-suited for handling big data with high velocity and volume.

In a big data environment, both relational and NoSQL databases can be used in conjunction with other big data technologies, such as Hadoop, Spark, and streaming platforms, to provide a comprehensive solution for big data processing and analytics. For example, relational databases can be used to store structured data and perform complex queries, while NoSQL databases can be used to store unstructured data and provide high-speed access. The choice of database technology depends on the specific use case and the nature of the data.

In conclusion, both types of databases can be used in conjunction with other big data technologies to provide a comprehensive solution for big data processing and analytics.

Part 2: b)

Choose your favourite NoSQL database. Write a brief report (600 words, ± 10%) summarising the history and usage of the tool, including key information such as the type of data model it utilises (key-value, document, wide-column or graph), and some particular use cases of the tool in industry, research or business.

My favorite NoSQL database is Apache Cassandra, a highly scalable and distributed NoSQL database that was originally developed at Facebook in 2008 and later released as an open-source project under the Apache License. Cassandra is designed to handle large volumes of data across multiple clusters and nodes, making it suitable for use cases that require high availability, fault tolerance, and linear scalability.

Cassandra utilizes a wide-column data model, also known as a column-family data model, which allows for efficient storage and retrieval of data in rows with multiple columns. This data model is similar to a traditional relational database but with some key differences. In Cassandra, columns are grouped together into column families, which are containers for rows that can have different column names and values. This flexibility in the data model allows for efficient handling of dynamic and changing data, making Cassandra well-suited for use cases that require storing and querying large amounts of semi-structured or unstructured data.

One of the key features of Cassandra is its ability to provide high availability and fault tolerance. Cassandra is designed to be distributed across multiple nodes, with data replicated across different nodes to ensure data durability and prevent data loss in case of node failures. Cassandra's distributed architecture allows it to scale horizontally, adding more nodes to the cluster as data volume and traffic increase, making it suitable for handling big data workloads.

Applications:

Financial Services: Cassandra is used by financial institutions for real-time data processing, fraud detection, risk management, and transactional data management. Its ability to handle high write loads and provide low-latency data access makes it well-suited for high-frequency trading, payment processing, and other financial applications that require high throughput and low-latency performance.

Internet of Things (IoT): Cassandra is used in IoT applications for storing and processing large volumes of sensor data, device telemetry, and time-series data. Its ability to scale horizontally and handle high write loads makes it suitable for capturing, storing, and analyzing data from millions of devices in real-time, enabling IoT applications such as smart cities, industrial automation, and remote monitoring.

E-commerce: Cassandra is used in e-commerce applications for handling large volumes of product catalog data, customer data, and transactional data. Its ability to provide high availability and fault tolerance ensures uninterrupted operation even during high-traffic periods, making it suitable for handling online retail and ecommerce workloads that require real-time data processing, personalized recommendations, and high-speed order processing.

Content Management: Cassandra is used in content management systems for storing and retrieving large volumes of unstructured or semi-structured data, such as multimedia content, social media data, and user-generated content. Its ability to handle complex data models and provide low-latency data access makes it suitable for content-driven applications that require fast and scalable data retrieval, such as digital media, social networking, and content delivery.

Healthcare: Cassandra is used in healthcare applications for storing and processing large volumes of patient data, medical records, and sensor data from medical devices. Its ability to provide high availability and scalability makes it suitable for handling real-time data processing, data analytics, and personalized medicine applications. In addition to these specific use cases, Cassandra is also used in research and academia for various data-intensive applications, such as scientific research, genomics, and machine learning.

Apache Cassandra has some limitations that users should be aware of, including its complexity due to its distributed architecture and data model, denormalized data modeling, eventual consistency, hardware requirements for running on commodity hardware, operational complexity, its own query language (CQL) that may require learning, optimization for write-heavy workloads, and trade-offs in CAP theorem for consistency.

Despite these limitations, Cassandra's ability to handle large-scale, high-velocity data workloads with high availability and fault tolerance has made it popular in various industries such as social media, e-commerce, finance, healthcare, and more. Understanding its limitations and leveraging its strengths can make Cassandra a powerful tool for building scalable and distributed data-driven applications.