Big Data: Frequently Asked Questions

Sudarshan Budhathoki

1. Define Big Data. What are its characteristics?

Big Data refers to extremely large, complex, and rapidly growing datasets that are difficult to capture, store, manage, and analyze using traditional data processing tools and systems. These datasets are generated from various sources such as social media, sensors, mobile applications, e-commerce platforms, and more.

Big Data is not just about the size of the data, but also about its structure, processing speed, accuracy, and value. The characteristics of Big Data are commonly referred to as the **5 V's**: Volume, Velocity, Variety, Veracity, and Value.

The 5 V's of Big Data

1. **Volume:**
   Refers to the massive scale of data generated every second. Traditional databases are unable to store and process this volume efficiently.
   Example: Facebook stores more than 4 petabytes of data per day.

2. **Velocity:**
   Describes the speed at which data is created, streamed, and analyzed. Real-time processing is essential in many applications.
   Example: Stock market data and IoT sensors generating continuous streams of data.

3. **Variety:**
   Refers to the different forms of data, including structured, semi-structured, and unstructured formats.
   Examples:

   - Structured: Tables in relational databases (e.g., MySQL)
   - Semi-structured: JSON, XML
   - Unstructured: Images, videos, emails, social media posts

4. **Veracity:**
   Indicates the trustworthiness and quality of data. Low veracity data may be inconsistent, noisy, or incomplete.
   Example: Inaccurate online reviews or duplicate sensor records.

5. **Value:**
   Represents the usefulness of data in making strategic decisions. Extracting value from raw data is the core objective of Big Data analytics.
   Example: E-commerce sites analyzing customer behavior to improve recommendations.

Summary Table: The 5 V's of Big Data

V	**Description**	**Example**
Volume	Huge quantity of data	Facebook storing petabytes daily
Velocity	Fast generation and processing	Streaming data from IoT devices
Variety	Different formats of data	Text, audio, video, JSON, SQL
Veracity	Accuracy and reliability	Noisy social media comments
Value	Insightful and useful outcomes	Predictive analytics in healthcare

Big Data: Frequently Asked Questions

Sudarshan Budhathoki

2. What is Hadoop? Explain its ecosystem.

Hadoop is an open-source software framework designed to store and process large-scale data in a distributed computing environment. It is capable of handling both structured and unstructured data efficiently. Hadoop's ecosystem allows horizontal scalability and fault tolerance through replication and parallelism.

Hadoop was originally developed by Doug Cutting and Mike Cafarella, and it is based on ideas from Google's File System (GFS) and MapReduce programming model.

Key Characteristics of Hadoop

- **Scalability:** Easily scales horizontally by adding commodity hardware.
- **Fault Tolerance:** Automatically replicates data to handle hardware failures.
- **Flexibility:** Can handle all types of data: structured, semi-structured, and unstructured.
- **Cost-effective:** Built on open-source software and inexpensive servers.

Major Components of Hadoop Ecosystem

1. **HDFS (Hadoop Distributed File System):**
   Stores large files across multiple machines with data replication.
   Example: A 2GB file may be split into 16 blocks of 128MB each.

2. **MapReduce:**
   Programming model for processing data in parallel using ”Map” and ”Reduce” steps.
   Example: Count word frequency in a large document corpus.

3. **YARN (Yet Another Resource Negotiator):**
   Resource management and job scheduling layer.
   Example: Assigning computing tasks across nodes.

4. **Hive:**
   Provides a SQL-like query language for Hadoop.
   Example: SELECT * FROM customers WHERE age ¿ 30;

5. **Pig:**
   Scripting language (Pig Latin) for analyzing large datasets.
   Example: LOAD and FILTER functions for log analysis.

6. **HBase:**
   A NoSQL database for real-time read/write access.
   Example: Storing time-series sensor data.

7. **Sqoop:**
   Transfers data between Hadoop and relational databases.
   Example: Importing sales data from MySQL to HDFS.

8. **Flume:**
   Collects, aggregates, and transports log data to Hadoop.
   Example: Streaming Apache logs into HDFS.

3. Describe the architecture of Hadoop.

Hadoop follows a master-slave architecture and consists of three main layers: storage, processing, and resource management.

1. Storage Layer (HDFS)

- **NameNode:** Stores metadata such as file names, permissions, and block locations.
- **DataNodes:** Store actual data blocks and respond to requests from the NameNode.
- **Replication:** Blocks are typically replicated 3 times for fault tolerance.

2. Processing Layer (MapReduce)

- **Map Task:** Processes input data and produces intermediate key-value pairs.
- **Shuffle and Sort:** Groups and sorts intermediate data by key.
- **Reduce Task:** Aggregates values and writes the output to HDFS.

3. Resource Management Layer (YARN)

- **ResourceManager:** Manages global resource allocation.
- **NodeManager:** Monitors and manages resources on individual nodes.
- **ApplicationMaster:** Manages execution of individual applications.

Example: In a video streaming platform, HDFS stores view logs, MapReduce aggregates view counts, and YARN schedules the tasks.

4. Explain how MapReduce works internally.

MapReduce is a parallel programming model used to process large-scale data across distributed systems.

Steps in MapReduce Processing

1. **Input Split:**
   Large input files are split into smaller fixed-size chunks (e.g., 128MB).
2. **Map Phase:**
   Each chunk is processed by a Mapper function that produces intermediate (key, value) pairs.
   Example: For text input, the Mapper emits (word, 1).
3. **Shuffle and Sort:**
   The system groups all values by their keys and sorts them before sending to Reducers.
4. **Reduce Phase:**
   Reducers combine the values for each key to produce final output.
   Example: Summing all counts of each word.
5. **Output:**
   Final results are written back to HDFS.

5. What are the advantages and limitations of Hadoop?

Advantages of Hadoop

1. **Scalability:**
   New nodes can be added easily without changing the data formats.
2. **Fault Tolerance:**
   Data is replicated to avoid loss during hardware failure.
3. **Flexibility:**
   Can handle any kind of data including text, video, audio, etc.
4. **Cost-Effective:**
   Works with low-cost hardware and is open-source.

Limitations of Hadoop

1. **Latency:**
   Hadoop is batch-oriented and not suitable for real-time analytics.
2. **Small Files Problem:**
   Large numbers of small files burden the NameNode.

3. **Complex Development:**
   Writing efficient MapReduce jobs requires deep knowledge.

4. **High Overhead:**
   MapReduce can be inefficient for certain data processing tasks.

Example: Hadoop is used by Facebook and LinkedIn for large-scale log analysis, but not for low-latency recommendation systems which use Spark or Flink.

Big Data: Frequently Asked Questions

Sudarshan Budhathoki

6. What are the advantages and limitations of Hadoop?

Advantages of Hadoop

1. **Scalability:**
   Easily scales horizontally across hundreds or thousands of commodity servers.
2. **Fault Tolerance:**
   Data is replicated across nodes. If one node fails, data is still available.
3. **Cost-Effectiveness:**
   Runs on low-cost hardware and is open-source, reducing infrastructure costs.
4. **Flexibility:**
   Can handle structured, semi-structured, and unstructured data.
5. **High Throughput:**
   Efficiently processes large volumes of data using parallel processing.

Limitations of Hadoop

1. **Latency:**
   Hadoop uses batch processing and is not suitable for real-time analytics.
2. **Small Files Problem:**
   Many small files overwhelm the NameNode and reduce efficiency.
3. **Complex Programming Model:**
   Writing MapReduce jobs is not straightforward and often requires Java expertise.
4. **Security Limitations:**
   Built-in security features are basic and may need third-party integration.
5. **Debugging Difficulty:**
   Troubleshooting distributed jobs is complex and time-consuming.

Example: Hadoop is great for analyzing logs in bulk but is not ideal for tasks needing real-time feedback like fraud detection.

7. Discuss the differences between traditional and Big Data analytics.

Traditional Analytics	Big Data Analytics
Works on structured data	Works on structured, semi-structured, and unstructured data
Data stored in RDBMS	Data stored in distributed file systems (e.g., HDFS)
Processing limited to a single machine or small clusters	Parallel processing across large clusters
Batch processing	Supports both batch and real-time processing
Limited scalability	High scalability with commodity hardware

Example: Traditional systems analyze sales in a single store; Big Data systems analyze global online transactions in real-time.

8. Write short notes on Sqoop and Flume.

Sqoop

Sqoop (SQL-to-Hadoop) is a tool designed to transfer data between Hadoop and relational databases.

Features:

- Efficiently imports data from MySQL, Oracle, etc. into HDFS or Hive.
- Can export data from Hadoop back to relational databases.

Example: Import sales data from a MySQL database into Hadoop for analysis.

Flume

Flume is a distributed service for collecting and transporting large volumes of streaming log data to HDFS.

Features:

- Designed for high-throughput and reliable data ingestion.
- Uses a simple architecture with sources, channels, and sinks.

Example: Stream live Twitter feed logs into HDFS for sentiment analysis.

9. Describe use cases of Big Data in various domains.

1. **Healthcare:**
   Analyzing patient records and genomic data for personalized treatment and disease prediction.

2. **Finance:**
   Real-time fraud detection and algorithmic trading using historical and live market data.

3. **Retail and E-Commerce:**
   Recommending products, optimizing inventory, and setting dynamic pricing.

4. **Telecommunications:**
   Predictive maintenance of network infrastructure and reducing customer churn.

5. **Social Media:**
   Sentiment analysis, targeted advertising, and trend prediction.

6. **Logistics and Transportation:**
   Route optimization, demand forecasting, and autonomous vehicle navigation.

Example: Amazon uses Big Data for personalized recommendations, while UPS uses it for delivery route optimization.

Big Data: Frequently Asked Questions

Sudarshan Budhathoki

10. What is Hadoop Distributed File System (HDFS)? Explain its architecture.

HDFS is the primary storage system of Hadoop. It is designed to store very large files reliably across machines in a large cluster. It provides high throughput access to application data and is suitable for large-scale data processing.

Key Features of HDFS

- **Fault Tolerance:** Data blocks are replicated across multiple nodes (default replication factor is 3).
- **Scalability:** Can scale by adding more commodity hardware nodes.
- **Write-once-read-many Model:** Data is typically written once and read multiple times.
- **High Throughput:** Optimized for large data batch processing.

Architecture

- **NameNode (Master):** Manages metadata like directory structure and file locations.
- **DataNodes (Slaves):** Store actual data blocks and perform read/write operations.
- **Secondary NameNode:** Periodically merges the edit logs with fsimage to assist the NameNode (not a backup).

Example: A 1GB file split into 8 blocks (128MB each), replicated 3 times, is distributed across multiple DataNodes.

11. What is MapReduce? Explain with an example.

MapReduce is a programming model for processing large-scale data across distributed clusters. It works in two main phases:

1. Map Phase

Takes input and converts it into a set of key-value pairs.

Example: Input: ”Hadoop is good. Hadoop is scalable.”
Mapper emits: (Hadoop, 1), (is, 1), (good, 1), (Hadoop, 1), (is, 1), (scalable, 1)

2. Shuffle and Sort Phase

System groups values by key and prepares data for reduction.

3. Reduce Phase

Aggregates values for each key.

Reducer Output: (Hadoop, 2), (is, 2), (good, 1), (scalable, 1)

Use Case: Word count, log aggregation, sentiment scoring.

12. Define data warehousing. How is it different from a database?

Data Warehousing is the process of collecting, storing, and managing large volumes of historical data from multiple sources for analysis and reporting.

Differences between Data Warehouse and Database

Database	**Data Warehouse**
Designed for real-time transactions	Designed for analytical processing
Handles current data	Stores historical data
OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
Normalized schema	Denormalized schema

Example: A database stores current customer orders; a warehouse stores 5 years of customer purchasing data for trend analysis.

13. What is YARN in Hadoop?

YARN (Yet Another Resource Negotiator) is the resource management and job scheduling layer in Hadoop 2.x and beyond. It separates the resource management layer from the processing layer.

Components of YARN

- **ResourceManager (RM):** Global master managing cluster resources and application submissions.

- **NodeManager (NM):** Monitors resource usage (CPU, memory) on each node and reports to the RM.
- **ApplicationMaster (AM):** Manages the execution of a single application.
- **Containers:** Allocated by RM and managed by NM to run tasks.

Example: In a Hadoop job, the ResourceManager assigns tasks, NodeManagers execute them in containers, and the ApplicationMaster coordinates their execution.

Big Data: Frequently Asked Questions

Sudarshan Budhathoki

14. Explain the CAP theorem with suitable examples.

CAP Theorem states that in any distributed data system, it is impossible to simultaneously guarantee all three of the following:

- **Consistency (C):** Every read receives the most recent write or an error.
- **Availability (A):** Every request receives a (non-error) response, without guarantee that it contains the most recent write.
- **Partition Tolerance (P):** The system continues to operate despite arbitrary partitioning due to network failures.

According to the theorem, a distributed system can only provide two of the three guarantees at any given time.

Examples:

- **CP System:** HBase – Consistent and partition-tolerant but may sacrifice availability during failures.
- **AP System:** Cassandra – Available and partition-tolerant but may return stale data.
- **CA System:** Typically found in traditional databases (non-distributed systems) – Consistent and available but not partition-tolerant.

15. What is NoSQL? Explain types of NoSQL databases.

NoSQL (Not Only SQL) refers to a broad class of databases that move away from the traditional relational model to provide flexible schemas, scalability, and performance on large, unstructured datasets.

Types of NoSQL Databases

1. **Document-based:**
   Store data in documents (e.g., JSON, BSON).
   Example: MongoDB.

2. **Key-Value Stores:**
   Store data as a collection of key-value pairs.
   Example: Redis, Riak.

3. **Column-oriented:**
   Store data in columns instead of rows, good for analytical queries.
   Example: Apache Cassandra, HBase.

4. **Graph-based:**
   Store relationships between data entities as graphs.
   Example: Neo4j.

Use Case: NoSQL databases are used in applications requiring scalability, availability, and flexible data models, such as real-time analytics and IoT.

16. Compare Hadoop 1.x and Hadoop 2.x.

Hadoop 1.x	**Hadoop 2.x**
Uses JobTracker and TaskTracker	Uses ResourceManager and NodeManager (YARN)
MapReduce is the only processing model	Supports other models (Spark, Tez, etc.)
Scalability is limited	Highly scalable and efficient
Single point of failure in JobTracker	Improved fault tolerance with YARN
Fixed slot configuration	Flexible resource allocation via containers

Example: Hadoop 2.x enables running multiple applications (e.g., Spark, Hive) on the same cluster, which Hadoop 1.x cannot.

17. Explain stream processing vs batch processing.

Batch Processing involves collecting data over time and processing it in bulk.

Stream Processing handles data in real-time as it arrives.

Comparison

Batch Processing	**Stream Processing**
Processes large data blocks periodically	Processes data in real-time
High latency	Low latency
Easier to implement	More complex infrastructure
Example: Hadoop MapReduce	Example: Apache Storm, Spark Streaming

Use Case: Banking systems use stream processing for fraud detection, while analytics systems may use batch processing for daily sales summaries.

Big Data: Frequently Asked Questions

Sudarshan Budhathoki

18. Define Flume. How does it help in data ingestion?

Apache Flume is a distributed, reliable, and available system for efficiently collecting, aggregating, and moving large amounts of log data from various sources to a centralized data store such as HDFS.

Key Features

- Handles high-volume streaming data.
- Ensures reliable data delivery using channel-based architecture.
- Supports multiple sources and sinks (e.g., syslogs, HDFS, HBase).

Flume Architecture

- **Source:** Accepts data (e.g., from a web server).
- **Channel:** Temporary store (e.g., memory, file-based).
- **Sink:** Outputs data to storage (e.g., HDFS).

Example: Streaming Apache web server logs to HDFS for real-time monitoring and analytics.

19. Describe any two Big Data processing frameworks.

1. Apache Spark

- In-memory computing engine.
- Supports batch processing, stream processing, machine learning, and graph processing.
- Faster than Hadoop MapReduce due to in-memory operations.

Example: Real-time analytics for user behavior tracking.

2. Apache Flink

- Stream-first distributed processing engine.
- Supports batch and real-time analytics.
- Provides exactly-once semantics and fault tolerance.

Example: Monitoring IoT sensor data in real-time.

20. Explain Pig and its components.

Apache Pig is a platform for analyzing large datasets using a high-level scripting language called Pig Latin.

Components of Pig

- **Pig Latin:** Procedural data flow language.
- **Grunt Shell:** Interactive shell for writing and executing Pig Latin scripts.
- **Execution Modes:**
  - Local Mode – for testing on local file system.
  - MapReduce Mode – runs on Hadoop cluster.
- **Pig Engine:** Translates Pig Latin scripts into MapReduce jobs.

Example:

```
data = LOAD 'input.txt' AS (word:chararray);
filtered = FILTER data BY word == 'hadoop';
DUMP filtered;
```

21. Explain Hive and its architecture.

Apache Hive is a data warehouse system built on top of Hadoop that allows querying and managing large datasets using a SQL-like language called HiveQL.

Hive Architecture

- **User Interface:** CLI, Web UI, or JDBC/ODBC connections.
- **Driver:** Manages session and executes HiveQL.
- **Compiler:** Converts HiveQL to execution plan.

- **Metastore:** Stores metadata (e.g., table schemas).
- **Execution Engine:** Executes the plan using Hadoop MapReduce or Spark.

Example Query:

```
SELECT COUNT(*) FROM sales WHERE region = 'Asia';
```

Use Case: Running analytical queries on large-scale log or transaction data.

Big Data: Frequently Asked Questions

Sudarshan Budhathoki

22. Discuss HBase and its use cases.

Apache HBase is a distributed, scalable, NoSQL database that runs on top of HDFS. It provides real-time read/write access to large datasets.

Key Features

- Column-oriented data model.
- Supports millions of rows and columns.
- Integrates with Hadoop and MapReduce.
- Provides random, real-time read/write access.

Use Cases

- Time-series data like sensor readings or log data.
- Real-time applications needing fast data retrieval.
- Storing social media feeds or clickstream data.

Example: Facebook's Messenger uses HBase to store messages for quick retrieval.

23. Explain the importance of data locality in Hadoop.

Data locality is a principle in Hadoop where computation is moved closer to where data resides, minimizing data transfer across the network.

Benefits

- Reduces network congestion.
- Improves job execution time.
- Enhances cluster performance and resource utilization.

Types of Data Locality

- **Node Local:** Task runs on the same node where data block exists.
- **Rack Local:** Task runs on a different node in the same rack.
- **Off-Rack:** Task runs on a node in a different rack (least efficient).

Example: A Map task processes a data block locally instead of fetching it from another machine over the network.

24. What is the role of NameNode and DataNode in HDFS?

NameNode (Master)

- Manages filesystem metadata (directories, file names, block locations).
- Keeps track of live DataNodes via heartbeats.
- Coordinates read/write operations between client and DataNodes.
- Critical single point of failure in non-HA setups.

DataNode (Slave)

- Stores actual data blocks.
- Responds to requests from NameNode.
- Periodically sends block reports and heartbeats to NameNode.

Example: In a 1GB file with replication factor 3, DataNodes store the blocks, and the NameNode tracks their locations.

25. Discuss Spark RDD and DataFrame APIs.

Resilient Distributed Dataset (RDD)

- Immutable distributed collection of objects.
- Offers low-level control over transformations and actions.
- Suitable for complex operations and custom functions.

Example:

```
val data = sc.textFile("input.txt")
val filtered = data.filter(line => line.contains("error"))
```

DataFrame API

- Distributed collection of data organized into named columns (like SQL tables).
- Optimized via Catalyst query optimizer and Tungsten execution engine.
- Easier to use and more efficient than RDDs for most use cases.

Example:

```
val df = spark.read.json("input.json")
df.filter($"age" > 30).show()
```

Comparison:

RDD	DataFrame
Low-level API	High-level abstraction
Less optimized	Optimized by Catalyst
More verbose code	Concise and readable syntax

Big Data: Frequently Asked Questions

Sudarshan Budhathoki

26. Describe the architecture of Apache Spark.

Apache Spark is a distributed in-memory data processing engine designed for fast computation. It supports batch processing, stream processing, machine learning, and graph processing.

Spark Architecture Components

- **Driver Program:** Runs the main function and creates SparkContext.
- **Cluster Manager:** Allocates resources (e.g., YARN, Mesos, Standalone).
- **Executors:** Run on worker nodes and execute tasks assigned by the driver.
- **Tasks:** Units of work sent to executors to process RDDs or DataFrames.

Example: In a Spark job, the driver schedules tasks, the cluster manager assigns resources, and the executors perform actual computation.

27. Explain data preprocessing in the context of Big Data.

Data preprocessing is the process of cleaning, transforming, and organizing raw data to make it suitable for analysis.

Steps in Big Data Preprocessing

1. **Data Cleaning:** Removing noise, handling missing values, correcting errors.
2. **Data Integration:** Combining data from multiple sources.
3. **Data Transformation:** Normalization, encoding, or aggregation of data.
4. **Data Reduction:** Reducing dimensionality or size (e.g., sampling, PCA).
5. **Data Discretization:** Converting continuous attributes into categorical.

Example: Cleaning customer transaction logs by removing duplicates and filling missing values.

28. What are the challenges of Big Data analytics?

1. **Data Volume:** Storing and processing petabytes of data efficiently.
2. **Data Variety:** Handling diverse formats like video, text, and images.
3. **Data Velocity:** Managing and analyzing high-speed data streams.
4. **Data Veracity:** Ensuring accuracy and trustworthiness of the data.
5. **Scalability:** Building infrastructure that scales with data growth.
6. **Security and Privacy:** Protecting sensitive data from breaches.
7. **Skilled Workforce:** Shortage of qualified professionals in big data technologies.

Example: Real-time fraud detection requires scalable and secure analytics that can process massive volumes of fast-moving financial data.

29. Discuss how Big Data is used in healthcare.

Applications in Healthcare

- **Predictive Analytics:** Forecasting disease outbreaks or patient readmission.
- **Personalized Medicine:** Tailoring treatment plans using genomic data.
- **Clinical Decision Support:** Assisting doctors with diagnosis via historical data.
- **Remote Patient Monitoring:** Using wearable devices to track patient vitals.
- **Healthcare Fraud Detection:** Identifying anomalous billing or claims.

Example: Using Apache Spark to analyze electronic health records (EHRs) for early detection of heart disease.

Big Data: Frequently Asked Questions

Sudarshan Budhathoki

30. Explain how Big Data is used in fraud detection.

Big Data enables real-time fraud detection by analyzing large volumes of transactions and identifying anomalies or patterns that may indicate fraudulent behavior.

Techniques Used

- **Machine Learning:** Trains models to detect unusual patterns in behavior.
- **Behavioral Analytics:** Builds profiles of normal user behavior to detect deviations.
- **Graph Analysis:** Identifies suspicious relationships between entities.
- **Stream Processing:** Enables real-time detection of anomalies as data flows in.

Example: Credit card companies use Big Data tools like Spark and Kafka to flag potentially fraudulent transactions in milliseconds.

31. Explain the role of Big Data in social media analytics.

Big Data plays a crucial role in analyzing the massive and dynamic information generated by social media platforms such as Twitter, Facebook, and Instagram.

Applications

- **Sentiment Analysis:** Analyzing user opinions and emotions about brands or events.
- **Trend Analysis:** Detecting viral topics and hashtags.
- **User Profiling:** Understanding user behavior for personalized recommendations.
- **Campaign Monitoring:** Measuring the reach and impact of marketing campaigns.
- **Influencer Detection:** Identifying key individuals with high engagement.

Example: Companies use NLP techniques on Twitter data streams to gauge public sentiment on product launches.