Unit V Big Data Processing in Distributed Systems

5.1 Big data processing frameworks in distributed computing: Hadoop, Apache Spark, Apache Storm, Samza, Flink

💾 Big Data Processing Frameworks in Distributed Computing

These frameworks help us **store, manage, and process large datasets** across multiple machines efficiently.

🐘 1. Hadoop

- **What it does:** Stores and processes large-scale data using a distributed file system (HDFS) and a processing model (MapReduce).
- **Batch processing**: Works well for jobs that analyze data in chunks (not real-time).

🧠 *Example:* Counting word frequency from a large collection of documents.

✅ **Good for:** Storage-heavy tasks, historical data analysis.
❌ **Not great for:** Real-time processing.

⚡ 2. Apache Spark

- **What it does:** Fast and powerful engine for **batch** and **streaming** data.
- Uses in-memory computing for faster performance than Hadoop.
- Has built-in libraries for ML (MLlib), SQL, Graph processing, etc.

🧠 *Example:* Running a recommendation system or fraud detection on live data.

✅ **Good for:** Fast processing, machine learning, complex analytics.
❌ **Needs more memory** compared to Hadoop.

🌩️ 3. Apache Storm

- **What it does:** Processes data in **real-time** using a topology of components (spouts and bolts).
- Good for continuous data streams.

🧠 *Example:* Analyzing Twitter data as it is being tweeted.

✅ **Good for:** Real-time analytics, low-latency systems.
❌ **Not built for batch processing.**

📊 4. Apache Samza

- **What it does:** Real-time data processing framework developed by LinkedIn.
- Works well with Apache Kafka for handling data streams.
- Focuses on **fault-tolerance and scalability**.

🧠 *Example:* Live user activity tracking on a website.

✅ **Good for:** Real-time event processing with strong integration with Kafka.

🌊 5. Apache Flink

- **What it does:** Processes both **real-time** and **batch** data.
- Known for **event-time processing**, **low latency**, and **stateful stream processing**.
- Can handle complex stream jobs.

🧠 *Example:* Fraud detection, real-time alerts in financial transactions.

✅ **Good for:** Complex streaming, exactly-once processing guarantees.
❌ **Can be overkill** for simpler jobs.

✅ **Summary Table**

Framework	Type	Use Case	Strength
Hadoop	Batch	Big data processing (offline)	Storage + simple processing
Spark	Batch + Streaming	Fast analytics, ML	Speed, versatility
Storm	Real-time	Live data stream analysis	Real-time, low latency
Samza	Real-time	Event-driven applications	Kafka integration, fault-tolerant
Flink	Batch + Real-time	Complex event processing	High accuracy, real-time + stateful

5.2 Parallel and distributed data processing techniques: Single Instruction Single Data (SISD), Multiple Instruction Single Data (MISD), Single Instruction Multiple Data (SIMD), Multiple Instruction Multiple Data (MIMD), Single program multiple data (SPMD), Massively parallel processing (MPP)

🧮 Parallel & Distributed Data Processing Techniques

These are different ways to **organize processing across one or more processors**, depending on how data and instructions are shared.

---

1. 🧍 SISD (Single Instruction Single Data)

- **One processor**
- Executes **one instruction** on **one piece of data** at a time.

🧠 *Example:* A traditional single-core CPU processing one task.

✅ Simple
❌ Slow for large data processing

---

2. 👯 MISD (Multiple Instruction Single Data)

- **Multiple processors**
- Each executes **different instructions** on the **same data**.

🧠 *Example:* Rare in practice, but used in **fault-tolerant systems** like aircraft control.

✅ Redundant safety checks
❌ Complex and uncommon

---

3. 🎯 SIMD (Single Instruction Multiple Data)

- **One instruction** applied to **multiple data points** at the same time.
- Used in **vector processing and GPUs**.

🧠 *Example:* Applying the same filter to all pixels in an image at once.

✅ Fast data processing
❌ All data must be processed the same way

---

4. 🧠 MIMD (Multiple Instruction Multiple Data)

- **Multiple processors**, each doing **different instructions** on **different data**.
- Most **modern CPUs, clusters, cloud systems** work this way.

🧠 *Example:* One processor sorts data while another runs a neural network.

✅ Very flexible
✅ Suitable for distributed systems

---

5. 💻 SPMD (Single Program Multiple Data)

- All processors run the **same program**, but on **different chunks of data**.
- Common in **parallel computing with MPI or GPUs**.

🧠 *Example:* Training a machine learning model where each processor trains on a different dataset batch.

✅ Easy to scale
❌ Coordination required between processors

---

6. 🚀 MPP (Massively Parallel Processing)

- **Hundreds or thousands of processors** work in parallel on parts of a big task.
- Often used in **data warehouses** and **big data platforms**.

🧠 *Example:* Analyzing huge retail datasets across multiple servers.

✅ High speed and scalability
✅ Ideal for big data analytics

✅ Summary Table

Technique	Instructions	Data	Where Used
SISD	Single	Single	Old CPUs, basic programs
MISD	Multiple	Single	Safety systems, rare use
SIMD	Single	Multiple	GPUs, image processing, scientific calc
MIMD	Multiple	Multiple	Most modern multi-core & distributed systems
SPMD	Same Program	Multiple Data	HPC, parallel scientific computing

Technique	Instructions	Data	Where Used
MPP	Many Processors	Distributed Data	Big data platforms, data warehouses

5.3 Scalable data ingestion: types of data ingestion, Benefits, challenges, tools, transformation in distributed systems

📥 Scalable Data Ingestion

Data ingestion is the process of **bringing data into a system** from various sources like databases, devices, apps, or sensors.

In **distributed systems**, it must be scalable so it can handle:

- Large volumes
- High velocity (speed)
- Diverse formats (structured, semi-structured, unstructured)

🔄 Types of Data Ingestion

1. **Batch Ingestion**
    - Data is collected and processed at **scheduled intervals**.
    - Good for large datasets that don't change often.

🧠 *Example:* Daily sales report from a retail store.

2. **Real-Time (Streaming) Ingestion**
    - Data is ingested **continuously as it arrives**.
    - Ideal for time-sensitive applications like fraud detection.

🧠 *Example:* Sensor data from IoT devices or social media streams.

3. **Lambda Architecture (Hybrid)**
    - Combines both batch and real-time processing.
    - Provides speed + accuracy.

✅ Benefits of Scalable Data Ingestion

- 💡 **Real-time insights**
- 🔗 **Integrates multiple sources** easily
- 📈 **Handles growth** in data volume or velocity
- ⚡ **Speeds up decision-making**
- 🤖 **Feeds data to AI/ML models efficiently**

⚠️ Challenges in Data Ingestion

Challenge	Description
⚡ High Data Velocity	Managing continuous inflow from millions of sources
🔀 Data Variety	Handling different formats (JSON, XML, images, logs, etc.)
💾 Storage Bottlenecks	Overloading storage or causing delays
⏳ Latency	Ensuring low-latency for real-time applications

Challenge	Description
🔐 Security & Compliance	Ensuring secure, private, and compliant data ingestion

---

🔧 Common Tools for Data Ingestion

Tool	Type	Use Case
Apache Kafka	Real-time / Stream	High-throughput event streaming
Apache Flume	Batch / Log data	Collecting logs from multiple sources
Apache NiFi	Batch + Stream	Visual data flow management
Logstash	Stream / Batch	ELK stack - log ingestion
Sqoop	Batch	Import/export between Hadoop & databases
Talend / Informatica	Both	Enterprise-level ETL and ingestion

🔄 Transformation in Distributed Systems

Before data is stored or analyzed, it is **transformed**:

- **Cleansing**: Removing duplicates, fixing missing values
- **Standardization**: Converting formats (e.g., date formats)

- **Aggregation**: Summarizing large data (e.g., average temperature per hour)
- **Enrichment**: Adding extra useful data (e.g., converting IP to location)

In distributed systems, these transformations are performed **in parallel** to save time.

🧠 Summary

Component	Explanation
Types	Batch, Real-time, Lambda (hybrid)
Benefits	Scalable, fast, real-time ready
Challenges	Volume, variety, latency, security
Tools	Kafka, NiFi, Flume, Logstash, Sqoop, Talend
Transformation	Cleansing, standardizing, enriching in parallel

5.4 Real-time analytics and Streaming analytics: types of real time analytics, types of streaming analytics, Comparison of real time analytics and streaming analytics, Applying AI and data science for large-scale data processing and analytics.

⏱️ Real-Time Analytics vs. Streaming Analytics

Both of these concepts involve processing data **as it arrives** (in contrast to batch processing), but they have key differences in how they handle the data and their use cases.

📊 Types of Real-Time Analytics

Real-time analytics focuses on analyzing data **immediately** after it is ingested, often with the goal of **providing insights or making decisions** instantly.

1. **Operational Analytics**
    - Monitors and analyzes operational data to detect issues or opportunities in real time.
    - **Use case:** Fraud detection in banking systems or monitoring server health.
2. **Descriptive Analytics**
    - Provides summaries of historical data as it flows in real-time.
    - **Use case:** Dashboards showing up-to-date metrics like daily sales or traffic.
3. **Predictive Analytics**
    - Uses real-time data and historical patterns to **predict future events**.
    - **Use case:** Predicting customer churn or stock market trends.
4. **Prescriptive Analytics**
    - Analyzes real-time data to **recommend actions**.
    - **Use case:** Real-time traffic routing recommendations for drivers.

📈 Types of Streaming Analytics

Streaming analytics is about processing **continuous streams of data** in real-time to extract actionable insights.

1. **Event Stream Processing (ESP)**
    - Processes **individual events** or **event sequences** as they happen.
    - **Use case:** Monitoring IoT devices, such as tracking sensor data from factory machines.
2. **Complex Event Processing (CEP)**
    - Analyzes **patterns of events** over time to identify more complex relationships or trends.
    - **Use case:** Fraud detection in financial transactions, where multiple transactions from the same user are analyzed together.
3. **Windowing**
    - Applies analytics to a **time window** of data, allowing users to analyze data that just arrived within a certain time frame.
    - **Use case:** Calculating moving averages or running totals.

🔄 Comparison: Real-Time vs. Streaming Analytics

Feature	Real-Time Analytics	Streaming Analytics
Focus	Analyzes data as it comes in (mostly for monitoring)	Continuous analysis of events over time

Feature	Real-Time Analytics	Streaming Analytics
Data Type	Static and flowing data	Continuous, flowing streams of data
Use Cases	Immediate action based on new data	Analyzing and detecting patterns across events
Latency	Often **low latency**, but depends on data frequency	Ultra-low latency, real-time processing
Examples	Real-time dashboards, operational monitoring	IoT data monitoring, fraud detection, clickstream analysis
Tools	Power BI, Tableau, Apache Kafka, AWS Kinesis	Apache Flink, Apache Storm, Apache Spark Streaming

---

🤖 Applying AI and Data Science for Large-Scale Data Processing and Analytics

AI and **data science** techniques are used to improve large-scale **real-time** and **streaming analytics**:

1. **Machine Learning for Predictive Analytics**
    - Machine learning models are trained using both **historical and real-time data** to make accurate predictions.
    - **Use case:** Predicting customer churn or product demand.
2. **Natural Language Processing (NLP)**
    - Analyzes **text data** in real-time, such as social media streams or customer reviews.
    - **Use case:** Sentiment analysis for brand monitoring.

3. **Deep Learning for Complex Event Detection**
    - Uses **deep neural networks** to identify patterns in large streams of data.
    - **Use case:** Real-time image or video analysis in security surveillance.
4. **Reinforcement Learning for Real-Time Decision Making**
    - Uses **feedback loops** to optimize decision-making in dynamic environments (e.g., autonomous driving, robotic systems).
    - **Use case:** Optimizing real-time bidding in advertising.
5. **Edge AI for IoT Data**
    - AI is applied directly at the **edge devices** (e.g., sensors, cameras) for real-time processing.
    - **Use case:** Analyzing data from sensors in smart factories for predictive maintenance.

✅ **Summary Table**

Aspect	Real-Time Analytics	Streaming Analytics
Processing Style	Processes incoming data immediately	Analyzes continuous streams of events over time
Focus	Instant insights and actions	Identifying complex patterns and trends in events

Aspect	Real-Time Analytics	Streaming Analytics
Data Type	Immediate, sometimes periodic data	Ongoing flow of data (continuous)
AI Applications	Predictive models, anomaly detection	Complex event processing, edge AI
Examples	Dashboard monitoring, fraud detection	IoT sensor data, real-time recommendation systems