Introduction to Big Data

The 5 V's and Why It Matters

Professor Anis Koubaa

SE 446
Alfaisal University

https://github.com/aniskoubaa/big_data_course

Spring 2026

Outline

Motivating Question

How much data is generated every minute?

Every 60 seconds:

- 500 hours of YouTube video uploaded
- 6 million Google searches
- 500,000 tweets posted
- 200 million emails sent

The Challenge:

- Too large for one machine
- Too fast for batch processing
- Too complex for simple queries
- Traditional DBs can't cope

Welcome to the Big Data Era

We need new tools and techniques to handle this scale!

What is Big Data?

Definition

Big Data refers to datasets that are too large, fast, or complex for traditional data processing tools.

- Cannot fit on a single machine
- Cannot be processed in reasonable time
- Requires **distributed computing**

Key Insight

It's not just about *size* — it's about the *challenges* of handling the data.

The Scale of Big Data

Company	**Data Generated**	**Scale**
Facebook	4 PB / day	250 billion photos
YouTube	500 hours video / minute	1 billion hours watched/day
Twitter	500 million tweets / day	6,000 tweets / second
Google	20 PB processed / day	3.5 billion searches / day

Perspective

1 Petabyte = 1,000 Terabytes = 1,000,000 Gigabytes

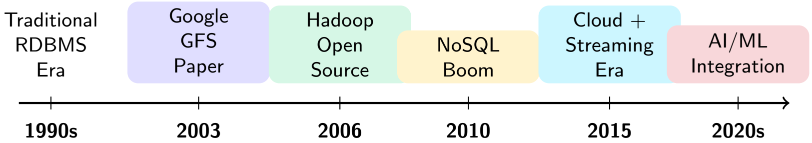
Big Data Use Cases & Industry Applications

Industry	Applications	Scale
Healthcare	Genomics, Patient Records, Drug Discovery, Epidemic Tracking	30+ PB human genome data
Finance	Fraud Detection, High-Frequency Trading, Credit Scoring, Anti-Money Laundering	Millions of transactions/sec
Smart Cities	Traffic Sensors, IoT Monitoring, Energy Grids, Pollution Tracking	Billions of sensor readings/day
Retail	Recommendation Engines, Demand Forecasting, Inventory Optimization	Amazon: 1.6M packages/day
Cybersecurity	Log Analysis, Threat Detection, Intrusion Prevention, SIEM	10TB+ logs/day in enterprises

Why This Matters

Every industry generates and depends on Big Data. The tools you learn in this course apply across all domains!

Historical Evolution of Big Data



Key Milestones:

- **2003**: Google GFS paper → distributed storage
- **2004**: Google MapReduce paper → parallel processing
- **2006**: Yahoo! releases Hadoop as open-source

Why This Matters:

- Hadoop was born from *real problems* at Google
- Open-source democratized Big Data
- Today: Cloud-native, real-time, AI-powered

The Modern Big Data Ecosystem

Big Data = Storage + Compute + Ingestion + Analytics + Visualization

Layer	Category	Technologies
Storage	Data Lakes Data Warehouses	S3, ADLS, MinIO, HDFS BigQuery, Snowflake, Redshift
Processing	Batch Stream Interactive Query	MapReduce, Spark Kafka, Flink, Storm Presto, Trino, Athena
NoSQL DBs	Key-Value Document Columnar Graph	Redis, DynamoDB MongoDB, Couchbase Cassandra, HBase Neo4j, Neptune
Analytics	BI / Visualization ML Platforms	Tableau, Looker, Power BI Spark MLlib, SageMaker

Course Focus

We focus on the **core**: HDFS, MapReduce, Hive, Spark, Kafka — the foundation for all the above.

Cloud & Modern Big Data Stack

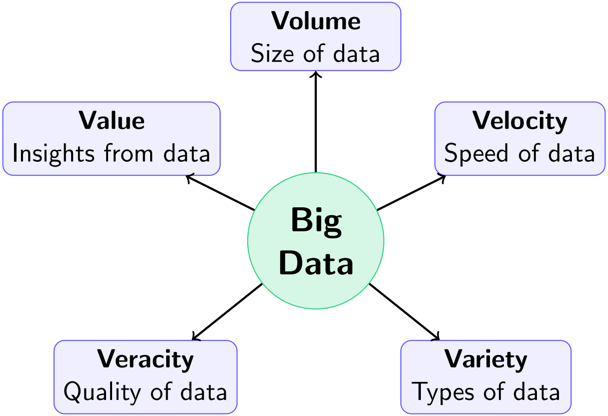
In 2026, Big Data is **cloud-native**. Key platforms you'll encounter:

Provider	**Category**	**Services**
AWS	Storage, Compute, ETL	S3, EMR (Hadoop/Spark), Glue, Kinesis, Redshift
GCP	Analytics, Streaming	BigQuery, Dataflow, Dataproc, Pub/Sub
Azure	Data Lake, Warehousing	ADLS, Synapse Analytics, HDInsight, Event Hubs
Databricks	Unified Analytics	Lakehouse, MLflow, Delta Lake, Spark (founders!)

Future-Proof Your Skills

The concepts you learn (HDFS, Spark, Kafka) translate directly to cloud equivalents. Databricks Community Edition (free) is our lab environment!

The 5 V's of Big Data



Volume: Size of Data

Challenge

Datasets too large to store or process on a single machine.

Examples:

- Genomic data: 100 GB per genome
- CERN: 1 PB / month
- Autonomous cars: 4 TB / day

Solution

Distributed Storage

- HDFS (Hadoop)
- Amazon S3
- Google Cloud Storage

Velocity: Speed of Data

Challenge

Data arrives too fast for batch processing.

Examples:

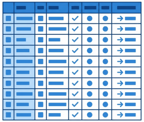
- Stock market: millions/second
- IoT sensors: continuous stream
- Social media: real-time feeds

Solution

Stream Processing

- Apache Kafka
- Spark Streaming
- Apache Flink

Structured vs. Unstructured Data



Structured Data

- **Definition**: Highly organized, fixed format.
- **Model**: Rows & Columns (RDBMS).
- **Examples**:
  - Excel spreadsheets
  - SQL Databases
  - Transaction logs
- **Search**: Easy to search.

Unstructured Data

- **Definition**: No tangible structure.
- **Model**: Binary large objects (BLOBs).
- **Examples**:
  - Images, Videos, Audio
  - Social Media Posts
  - PDFs, Emails
- **Search**: Requires advanced AI/ML.

Key Stats: Unstructured data accounts for $>$ 80% of all enterprise data!

Semi-Structured Data: The Missing Middle

Definition

Data with **some organization** but not fixed rows/columns. Self-describing with tags, keys, or markers.

Characteristics:

- Flexible, evolving schema
- Human & machine readable
- Hierarchical or nested structure
- No strict RDBMS constraints

Common Examples

- **JSON**: APIs, NoSQL (MongoDB)
- **XML**: Configuration, legacy systems
- **CSV**: Evolving headers, mixed types
- **Logs**: Apache, Nginx, application
- **Sensor Data**: IoT payloads

Why It Matters

80% of Kafka + IoT + API data is semi-structured! Tools like Spark and Hive have native support for JSON/XML parsing.

Big Data File Formats: Row vs. Columnar

Row-Based Formats

- **Examples**: CSV, JSON, Avro
- **Storage**: Row 1, Row 2, Row 3...
- **Best For**: Insert-heavy, full-row access
- **Limitation**: Read entire row even for 1 column

Columnar Formats

- **Examples**: Parquet, ORC
- **Storage**: Col A, Col A, Col A... then Col B...
- **Best For**: Analytics (SELECT specific columns)
- **Benefit**: Read only needed columns $\rightarrow$ 10x faster!

Format	Type	Compression	Use Case
CSV/JSON	Row	Poor	Interchange, APIs
Avro	Row	Good	Kafka, streaming
Parquet	Columnar	Excellent	Spark, Hive, BigQuery
ORC	Columnar	Excellent	Hive (optimized)

Industry standard: Store raw data in Data Lake, convert to Parquet for analytics!

Row vs. Columnar: Visual Example

Sample Data Table (3 rows, 3 columns)

Name	Age	City
Alice	25	Riyadh
Bob	30	Jeddah
Carol	28	Dammam

Row Storage (CSV)

```
Alice, 25, Riyadh
Bob, 30, Jeddah
Carol, 28, Dammam
```

Query `SELECT Age`: Read ALL data!

Columnar Storage (Parquet)

Name: `Alice, Bob, Carol`
Age: `25, 30, 28`
City: `Riyadh, Jeddah, Dammam`

Query `SELECT Age`: Read Age column only!

Result: Columnar reads 33% of data vs 100% for row-based → 3x faster!

Veracity & Value

Veracity: Data Quality

- Missing values
- Inconsistent formats
- Noise and outliers
- Fake data (bots, spam)

"Garbage in, garbage out"

Value: Extracting Insights

- Predictive analytics
- Customer segmentation
- Fraud detection
- Recommendation engines

"The goal of Big Data"

Limitations of RDBMS

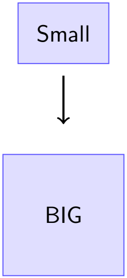
Challenge	RDBMS	Big Data
Scaling	Vertical (bigger server)	Horizontal (add nodes)
Schema	Fixed, predefined	Flexible, schema-on-read
Data Types	Structured only	All types
Cost	Expensive hardware	Commodity hardware
Speed	Slow for massive writes	Parallel distributed writes

The Solution

Distributed Systems: Hadoop, Spark, NoSQL databases

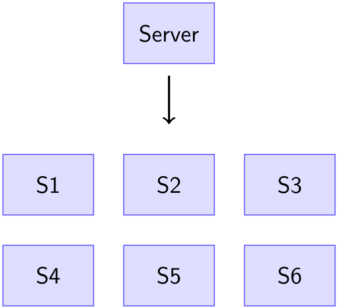
Vertical vs. Horizontal Scaling

Vertical Scaling



Expensive, limited

Horizontal Scaling



Scalable, cost-effective

Vertical Scaling vs. Horizontal Scaling (Deep Dive)

Vertical Scaling (Scale Up)

Increasing the capacity of a SINGLE machine (e.g., adding more RAM, stronger CPU).

Horizontal Scaling (Scale Out)

Adding MORE machines (nodes) to the system to work as a single cluster.

Feature	**Vertical Scaling**	**Horizontal Scaling**
Cost	High (Exponential)	Low (Commodity hardware)
Complexity	Low (Single System)	High (Distributed System)
Fault Tolerance	Single Point of Failure	High (Replication)
Downtime	Required for upgrades	Zero (Add nodes live)
Limit	Hardware Ceiling	Virtually Unlimited

The Secret Sauce: Data Locality

Traditional Approach

Move Data to Code

Query → Fetch 1 PB over network → Process

Problem: Network = Bottleneck!

Big Data Approach

Move Code to Data

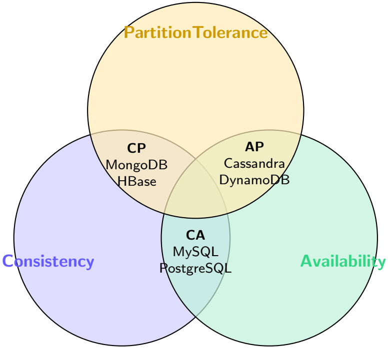
Send tiny Spark/MR task → Run locally on each node

Result: Parallelism, no network flood!

Metric	Traditional	Data Locality
Network Traffic	1 PB (entire dataset)	<1 MB (code only)
Processing Speed	Hours/Days	Minutes
Scalability	Limited by bandwidth	Scales with nodes

This is why HDFS stores data across nodes — so Spark/MapReduce can process locally!

The Theoretical Limit: CAP Theorem



Eric Brewer's Theorem (2000)

In a distributed system, you can only guarantee **two** of three properties:

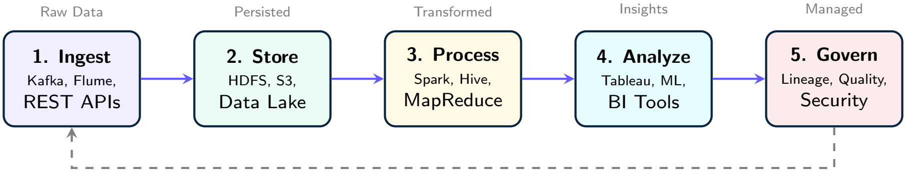
- **C (Consistency)**: All nodes see the same data at the same time.
- **A (Availability)**: Every request gets a response (even if stale).
- **P (Partition Tolerance)**: System works despite network failures.

Why This Matters

Big Data systems (Hadoop, NoSQL) prioritize **P** (network failures are inevitable), forcing a trade-off between C and A.

(Advanced topic — don't worry if you don't fully grasp it yet. We'll revisit this when covering NoSQL.)

The Big Data Pipeline Lifecycle



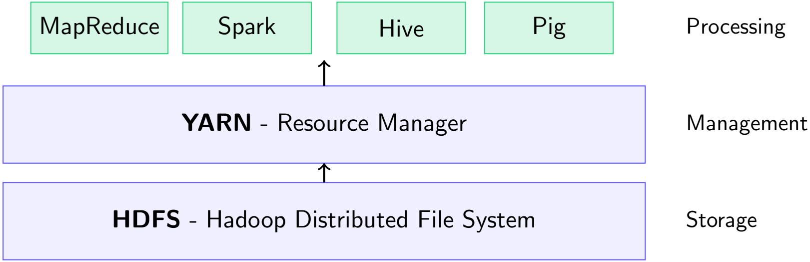
Processing Paradigms

- **Batch**: Daily/hourly (Spark, MapReduce)
- **Stream**: Real-time (Kafka, Flink)
- **Interactive**: Ad-hoc SQL (Presto)

ETL vs ELT

- **ETL**: Extract → Transform → Load (Data Warehouse)
- **ELT**: Load raw first, transform later (Data Lake)

The Hadoop Ecosystem



This Course Covers

HDFS, MapReduce, Hive, Spark, Kafka (Streaming)

Big Data Career Paths

Learning Big Data opens doors to multiple career paths:

Data Engineer

- Build pipelines
- Manage storage (HDFS, S3)
- Spark, Kafka, Airflow

Avg: $130K/year

Data Scientist

- ML on Big Data
- Statistical analysis
- Python, Spark MLlib

Avg: $140K/year

Data Analyst

- Dashboards & BI
- SQL, Hive, Presto
- Tableau, Power BI

Avg: $85K/year

This course gives you foundational skills for **all three paths**!

Summary: Key Takeaways

1. **Big Data** = Volume + Velocity + Variety + Veracity + Value
2. **Three data types**: Structured, Semi-structured, Unstructured
3. **RDBMS limitations** solved by distributed systems
4. **Hadoop Ecosystem**: HDFS (storage), YARN (resources), Spark/Hive (processing)

Next Session

HDFS Architecture: NameNode, DataNode, Replication

Homework

1. Watch the pre-class video for Session 2B:
   - "HDFS Tutorial" - Edureka (20 min)
2. Setup your accounts (if you haven't):
   - Google Colab: `colab.google.com`
   - GitHub: `github.com`
3. Review the notebook from today's session

Questions?

Prof. Anis Koubaa
akoubaa@alfaisal.edu