DSECL ZG 522: Big Data Systems

Session 1.1: Introduction to Big Data

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

25-October-2025

Janardhanan PS
janardhanan.ps@wilp.bits-pilani.ac.in

What to expect from this course

- The course introduces **Systems** for data analytics with particular emphasis on **Storage** and **Processing** of Big Data.
- It starts with computing models for distributed processing for scalability and fault-tolerance.
- It covers frameworks/tools for data **Ingestion** in to distributed file systems, and **Batch processing**, in-memory **Distributed processing** and **Stream processing** for real time analytics.
- NoSQL databases are covered in detail.
- This course will help students to master **essential skills** on NoSQL databases, Hadoop eco-system products, and Apache Spark framework including Spark SQL, Spark Streaming and Machine learning programming.
- Amazon's storage and database services are used as exemplar platforms on Cloud.

How Data Scientists find this course beneficial

- Data scientists use machine learning and predictive analytics to gain insights from large amounts of data (bigdata).
- To prepare the data, you should build expertise in big data platforms and tools, including NoSQL databases, Hadoop, MapReduce, Pig, Hive, and Spark.
- It would be helpful if you are fluent in any programming language (Java, Python, Scala), and Structured Query Language (SQL).
- Acquire working level knowledge on Linux Operating system
- Spend some time in learning descriptive and inferential statistics

Learn about AI agents for data engineering and data science

Topics for Situated / Experiential learning

Topic No.	Selected Topics in Syllabus for experiential learning
1	• Exercises on Distributed Systems – Hadoop configuration and HDFS • Exercises using Map-Reduce model: Standard patterns in map reduce models.
2	• Exercises on NoSQL – Installation and configuration • Exercises with NoSQL database – Simple CRUD operations and Failure / Consistency tests; • Cassandra - Consistency levels, CRUD operations, Schema on read, queries • MongoDB – installation, data ingestion, queries • Neo4j Graph database - Relationships and queries • HBase queries
3	• Exercises with Pig queries to perform Map-reduce job and understand how to build queries and underlying principles; • Exercises on creating Hive databases and HiveQL query operations, exploring built in functions, partitioning, data analysis
4	• Exercises on Spark to demonstrate RDD, and operations such as Map, FlatMap, Filter, PairRDD; • Typical Spark Programming idioms such as : Selecting Top N, Sorting, and Joins; • Exercises on DataFrames, Datasets, and Spark SQL • Spark Streaming - Sample Streams, Structured Streaming, Windowed Streaming
5	• Exercises using Spark MLlib: Regression, Classification, Collaborative Filtering, Clustering
6	• Exercises on Analytics on the Cloud – using AWS S3, AWS EMR, AWS data stores / databases, Querying with DynamoDB.

Text Books, References

T1	Seema Acharya and Subhashini Chellappan. *Big Data and Analytics*. Wiley India Pvt. Ltd. Second Edition
T2	Raj Kamal and Preeti Saxena, *Big Data Analytics*. McGraw Hill Education (India) Pvt.Ltd
R1	DT Editorial Services. *Big Data - Black Book*. DreamTech. Press. 2016
R2	Kai Hwang, Jack Dongarra, and Geoffrey C. Fox. *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Morgan Kauffman 2011
R3	Martin Kleppmann - Designing Data-Intensive Applications - O'Reilly, 2017
AR	Additional reading (As per Topic)

https://www.odbms.org/ - Resource Portal for AI, Gen AI, Big Data, Data Science, Data Management Technologies

Topics - Sessions

- S1: Introduction to Big Data and data locality
- S2: Parallel and Distributed Processing
- S3: Big Data Analytics and Big Data System characteristics
- S4: Consistency, Availability, Partition tolerance and Data Lifecycle
- S5: Typical NoSQL Databases
- S6: Big Data Lifecycle and distributed computing
- S7: Hadoop Architecture and Programming
- S8: Hadoop subsystems for Storage and Processing

  Mid-Sem Exam
- S9-S10: Hadoop ecosystem technologies
- S11-14: In-memory computing and streaming - Spark
- S15: Big Data and Cloud Computing
- S16: Big Data Storage on cloud (AWS)

Topics for today

- **Motivation**
  - ✓ **Why do modern Enterprises need to work with volume data**
  - ✓ **What is Big Data and data classification**
  - ✓ **Scaling RDBMS**
- What is a Big Data System
  - ✓ Desirable characteristics
  - ✓ Design challenges
- Architecture
  - ✓ High level architecture of Big Data solutions
  - ✓ Technology ecosystem
  - ✓ Case studies

Example of a data-driven Enterprise: A large online retailer (1)

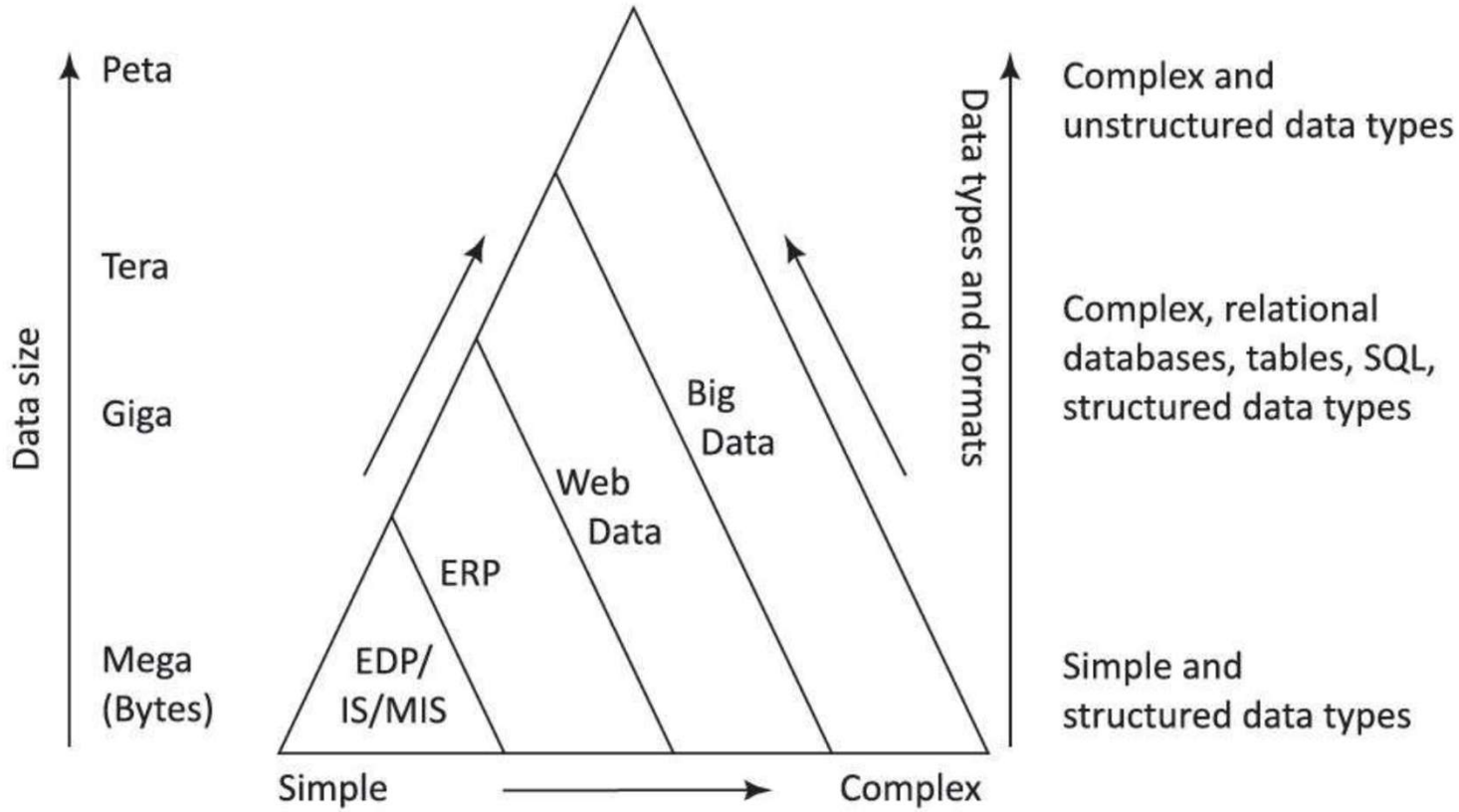
- What data is collected
  - ✓ Millions of transactions and browsing clicks per day across products, users
  - ✓ Delivery tracking
  - ✓ Reviews on multiple channels - website, social media, customer support
  - ✓ Support emails, logged calls
  - ✓ Ad click and browsing data
  - ✓ …
- Data is a mix of metrics, natural language text, logs, events, videos, images etc.

Example of a data-driven Enterprise: A large online retailer (2)

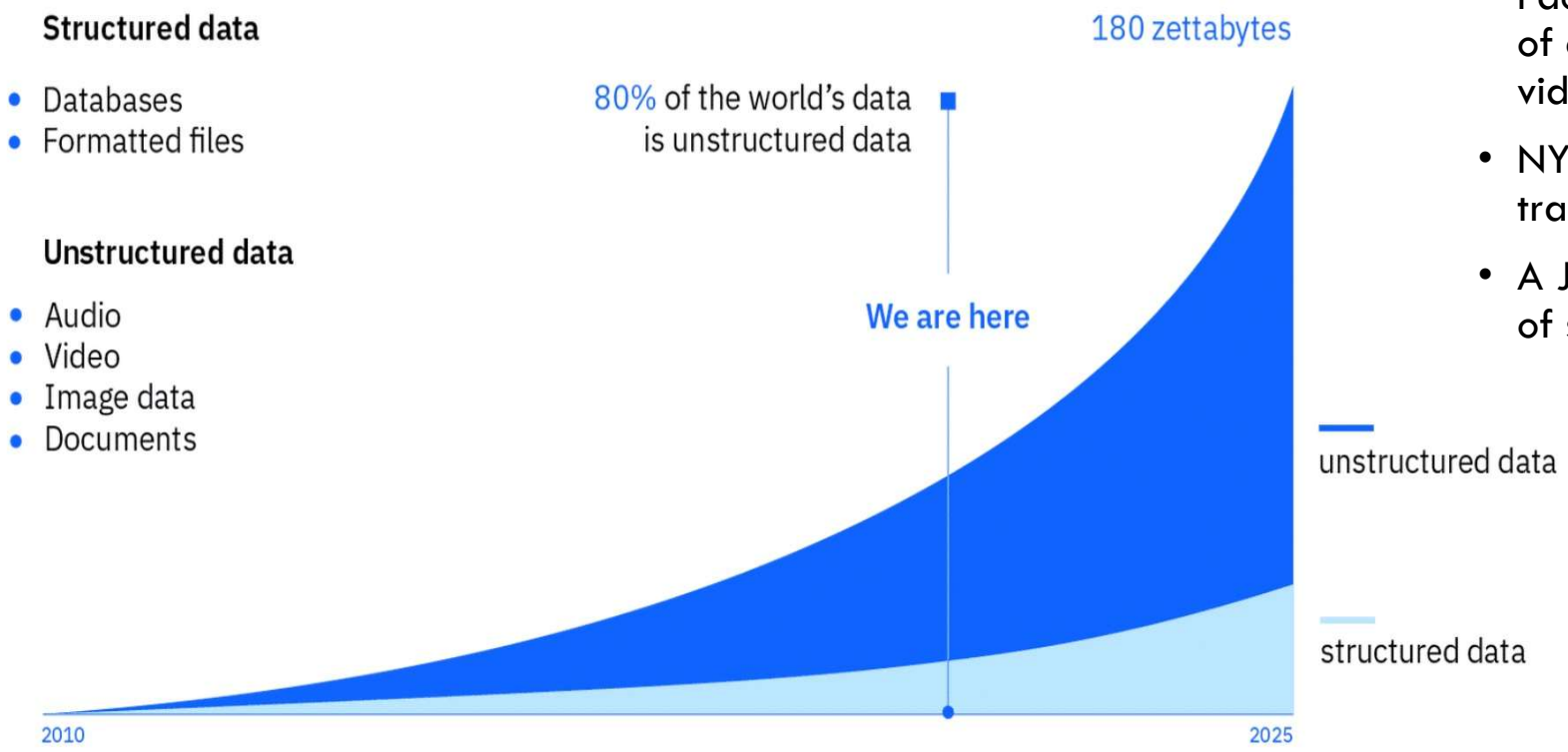
- What is this data used for
  - ✓ User profiling for better shopping experience
  - ✓ Operations efficiency metrics
  - ✓ Improve customer support experience, support training
  - ✓ Demand forecasting
  - ✓ Product marketing
  - ✓ …
- Data is the only way to create competitive differentiators, retain customers and ensure growth

Evolution of Bigdata and their characteristics

Kilobyte (KB) – 10^3 bytes
Megabyte (MB) – 10^6 bytes
Gigabyte (GB) – 10^9 bytes
Terabyte (TB) – 10^{12} bytes
Petabyte (PB) – 10^{15} bytes
Exabyte (EB) – 10^{18} bytes
Zettabyte (ZB) – 10^{21} bytes
Yottabyte (YB) – 10^{24} bytes



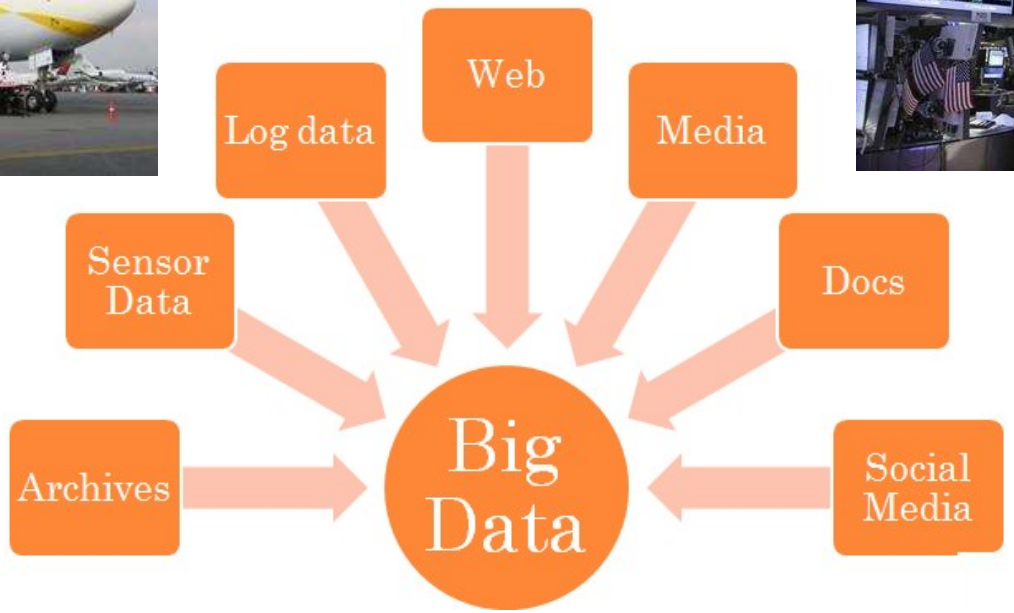
Data volume growth



- Facebook: 500+ TB/day of comments, images, videos etc.
- NYSE: 1TB/day of trading data
- A Jet Engine: 20TB / hour of sensor / log data

Source : What is big data?

Variety of data sources



Source : What is Big Data?

Big Data Characteristics

- How big is the Big Data?
- What is big data today may not be so tomorrow
- One's Big Data may be small Data for another

Any data that can challenge the current technology in some manner can be considered as Big Data:

- Storage
- Communication
- Speed of Generating
- Meaningful Analysis

Big Data Challenge

- In the past, the most difficult problem for businesses was how to store all the data.
- The challenge now is no longer to store large amounts of information, but to understand and analyze this data.
- By making sense out of this data through sophisticated analytics, and by presenting the key findings in an easily discernable fashion, we can derive value out of Big data
- Big data creates a new **Digital divide** – Those who can process and those who cannot
- **Data is only as useful as the decisions it enables**

Evolution of Web applications

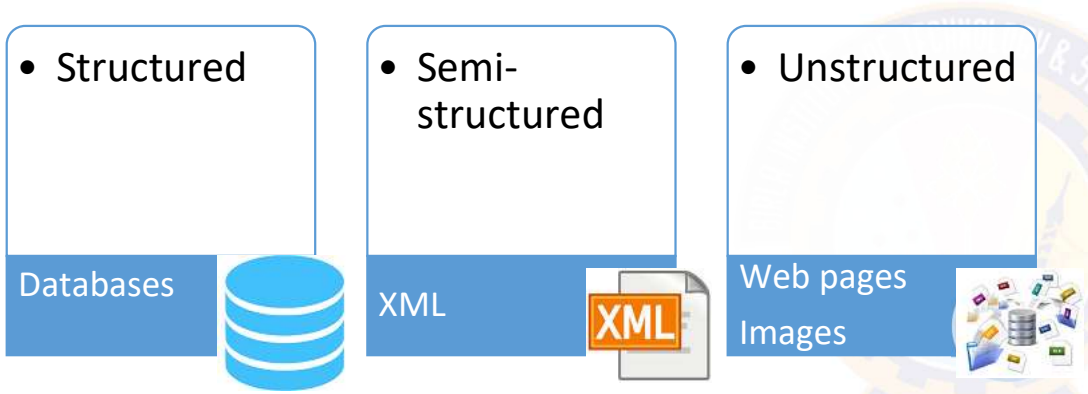
- Online systems of 1970s were designed for direct human interaction using direct attached terminals
- Interactive software has undergone fundamental changes over last 35 years
- They **evolved** into today's web and mobile applications
- Now these systems need to handle millions of concurrent interactive users
- The architecture of software systems has also transformed drastically

Changes in Data used in Web applications

- Web applications generate lot of temporary data that do not really belong to the main structured data store. Eg:
  - ✓ shopping carts
  - ✓ retained searches
  - ✓ site personalization
  - ✓ incomplete user questionnaires.
- Data set consists of large quantities of unstructured data in the form of Text,Images,Videos etc.
- Binary large objects in RDBMS (BLOB, CLOB) cannot handle this properly.
- Local data transactions that do not have to be very durable. Eg: – "liking" items on website
- Need to run queries against data that do not involve simple hierarchical relations

  Eg: "all people in a social network who have not purchased book this year "

Data classification



- Structured data is metrics, events that can be put in RDBMS with fixed schema
- Semi-structured data are XML, JSON structure where traditional RDBMS have support with varying efficiency but needs new kind of NoSQL databases
- New applications produce unstructured data which could be natural language text and multi-media content

Data usage pattern

- Higher demand now of analyzing unstructured data to glean insights
- Examples:
  - ✓ Analysis of social media content for sentiment analysis
  - ✓ Analysis of unstructured text content by search engines on the web as well as within enterprise
  - ✓ Analysis of support emails, calls for estimating customer satisfaction
  - ✓ NLP / Conversational AI for answering user questions from backend data

Structured Data

- Data is transformed and stored as per pre-defined schema
- Traditionally stored in RDBMS
- CRUD operations on records
- ACID semantics (Atomicity, Consistency, Isolation, Durability)
- Fine grain security and authorisation
- Known techniques on scaling RDBMS - more on this later
- Typically used by Systems of Record, e.g. OLTP systems, with strong consistency requirements and read / write workloads

```
TABLE Employee (
   emp_id int PRIMARY KEY,
   name varchar (50),
   designation varchar(25),
   salary int,
   dept_code int FOREIGN KEY
)
```

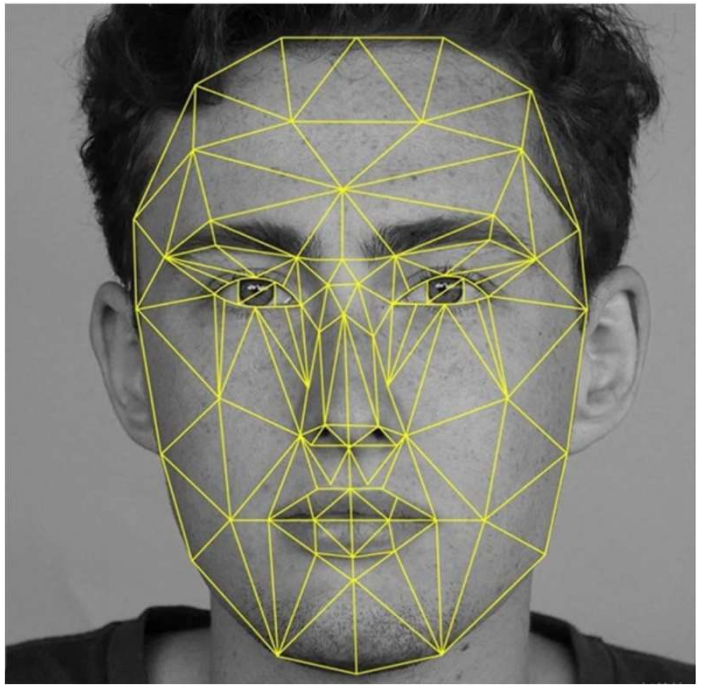
Semi-Structured Data

- No explicit data and schema separation
- Models real life situations better because attributes for every record could be different
- Easy to add new attributes
- XML, JSON structures
- Databases typically support flexible ACID properties, esp consistency of replicas
- Typically used by Systems of Engagement, e.g. social media

```
{
  "title": "Sweet fresh strawberry",
  "type": "fruit",
  "description": "Sweet fresh strawberry",
  “image": "1.jpg",
  “weight": 250,
  "expiry": 30/5/2021,
  "price": 29.45,
  “avg_rating": 4
  “reviews”: [
  { “user” : “p1“,  “rating”: 2, “review”: “ ….. “
}
```

Unstructured Data (1)

- More real-life data
  - ✓ video, voice, text, emails, chats, comments, reviews, blogs …
- There is some structure that is typically extracted from the data depending on the use case
  - ✓ image adjustments at pixel structure level
  - ✓ face recognition from video
  - ✓ tagging of features extracted in image / video
  - ✓ annotation of text



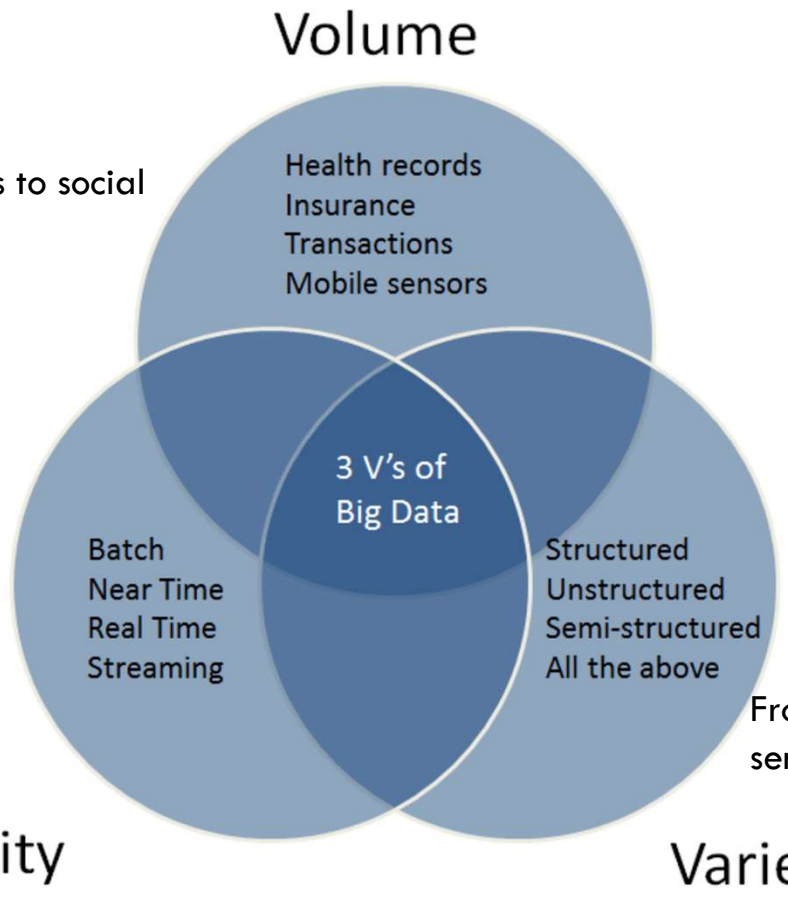
Unstructured Data (2)

- What can we do with it ?
  - ✓ Data mining
    - Association rule mining, e.g. market basket or affinity analysis
    - Regression, e.g. predict dependent variable from independent variables
    - Collaborative filtering, e.g. predict a user preference from group preferences
  - ✓ NLP - e.g. Human to Machine interaction, conversational systems
  - ✓ Text Analytics - e.g. sentiment analysis, search
  - ✓ Noisy text analytics - e.g. spell correction, speech to text

Define Big Data – 3Vs and beyond

From patient records to social media

From modelling jobs to fraud detection

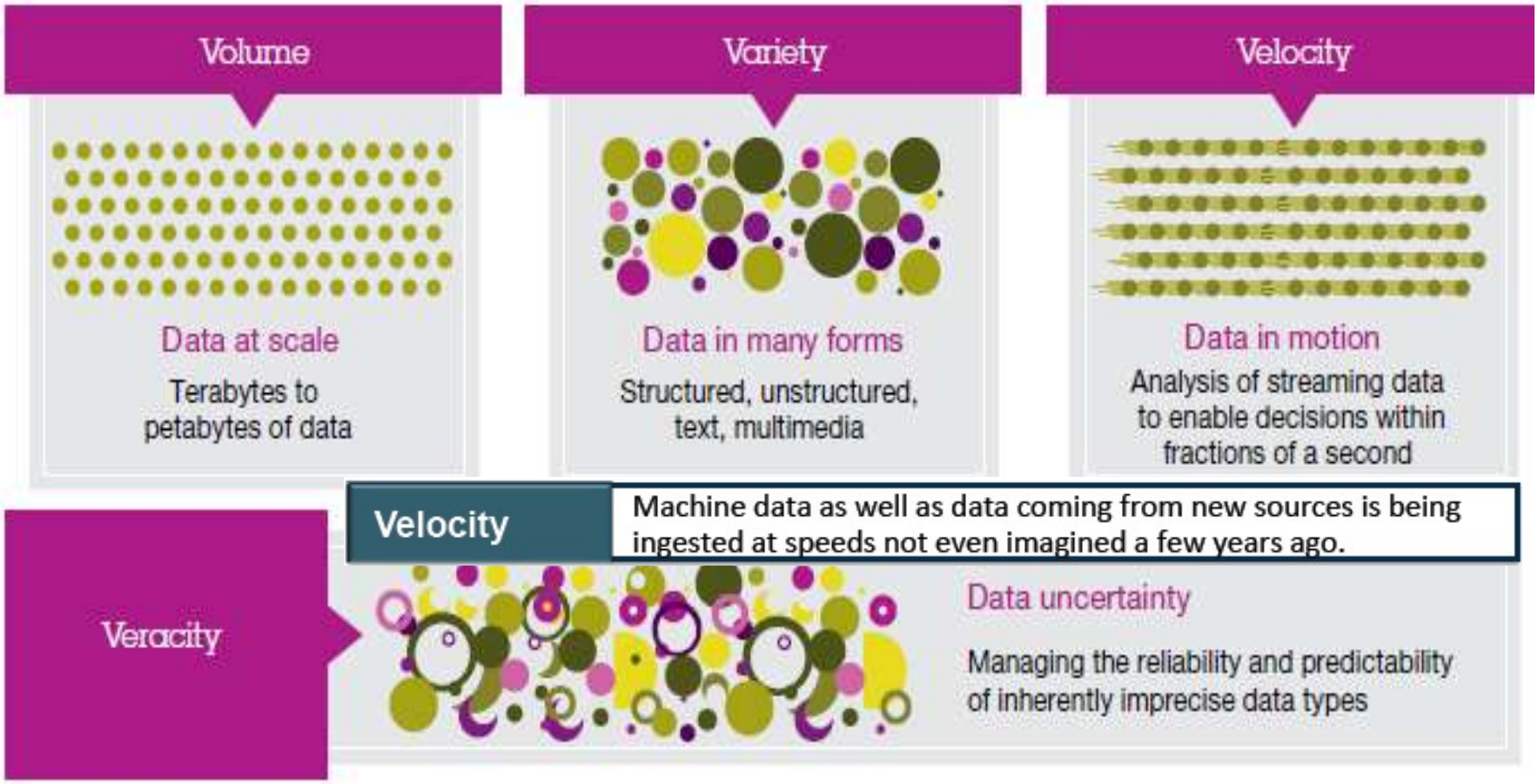


From clickstream analysis to sentiment analysis

Some make it 4Vs

Volume	Velocity	Variety	Veracity
Terabytes to Exabytes of Existing Data to be processed **Data at Rest**	Streaming Data, Milliseconds to seconds to respond **Data in Motion**	Structured, Unstructured, text, Multimedia **Data in many forms**	Uncertainity due to data inconsistency, incompleteness, ambiguity, latency, deception etc **Data in Doubt**

Big Data – More Vs

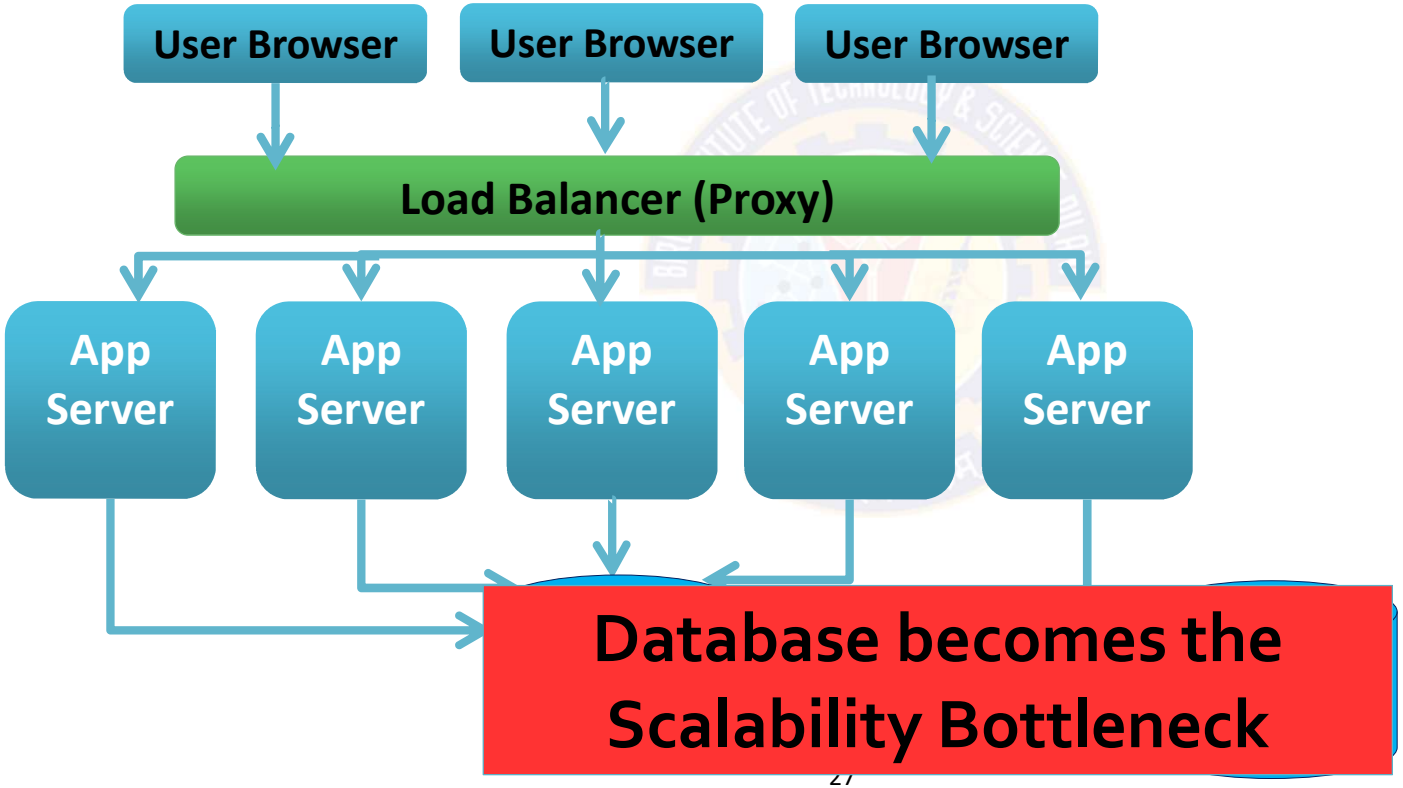


Velocity – Data streams

- What are Data Streams?
  - Continuous streams
  - Huge, Fast, and Changing
  - Scan the data only once
- Why Data Streams?
  - The arriving speed of streams and the huge amount of data are beyond our capability to store them.
  - "Real-time" processing
- Window Models
  - Landscape window (Entire Data Stream)
  - Sliding Window
  - Hopping Window
- Mining Data Stream

Scalability of Web applications

- Need to handle 1 to 1 million concurrent users
- Applications should provide uniform response



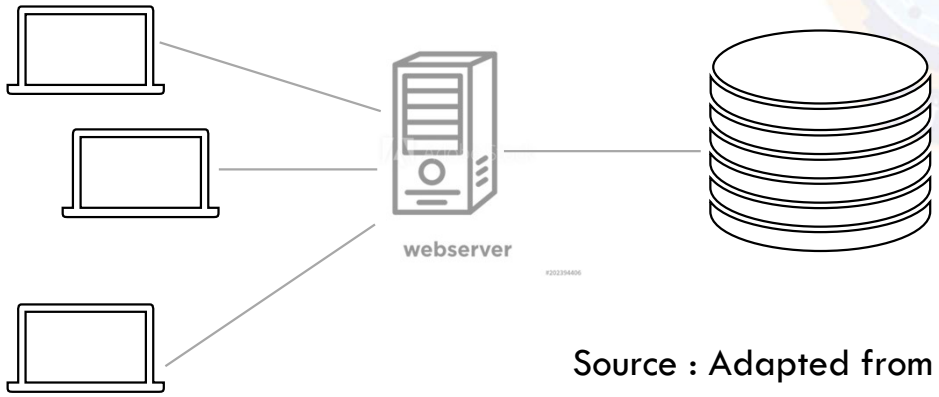
RDBMS and Web applications

- Most enterprise solutions have RDBMS back-end
- Very little change to RDBMS between 1980 and 2000
- Many application servers, one database:
  - ✓ Response slows down when DB is concurrently accessed by applications and Analytics tools
  - ✓ Response of SQL queries depend on size of table in the data base
  - ✓ Easy to parallelize application servers to 100s of servers, but
  - ✓ Harder to parallelize databases to same scale
- Most data originates from devices and volume of data grows at very high rate
- Web-based applications shows spikes in usage
  - Especially true for public-facing e-Commerce sites
- RDBMS becomes a point of contention

Isn't a traditional RDBMS good enough ?

Example Web Analytics Application

- Designing an application to monitor the page hits for a portal
- Every time a user visiting a portal page in browser, the server side keeps track of that visit
- Maintains a simple database table that holds information about each page hit
- If user visits the same page again, the page hit count is increased by one
- Uses this information for doing analysis of popular pages among the users



Column	Data Type
Id	Integer
User_ID	Integer
Page_URL	Varchar
Page_count	Long

Source : Adapted from Big Data by Nathan Marz

RDBMS is optimized for space- Not for speed

- Normalization removes data duplication and ensures data consistency
- RDBMS table schemas are highly normalized to minimize the data storage and to speed up inserts, update and deletes
- High degree of normalization is a disadvantage when it comes to retrieving data, as multiple tables may have to be joined to get all the desired information
- Creating these joins and reading from multiple tables can have a severe impact on performance, as multiple reads to disk may be required
- Analytical queries need to access large portions of the whole database, resulting in long run times

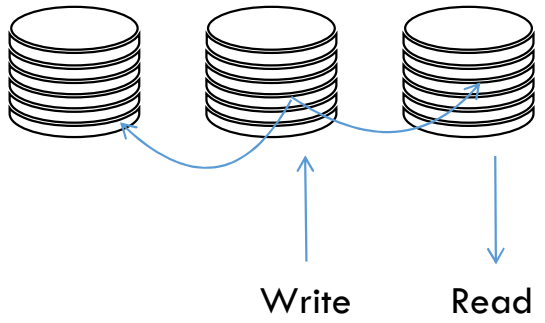
Fixed Table - No flexibility

- RDBMS Tables are designed and fixed once for all
- Number of columns in a table cannot be changed without stopping a running application
- Schema definition of tables cannot be changed on the fly
- Any change to the table definition involves recreation of the table lasting for several hours/days
- Growth of number of rows in a table is not unlimited.
- Time taken for processing queries varies with size of table
- Application performance degrades as the number of rows in a table increase (even with indexing)
- For example, aggregation of values in a table of 1 billion entries may take hours together

Issues with RDBMS (1)

- Not all BigData use cases need strong ACID semantics, esp Systems of Engagement
  - ✓ ACID becomes a bottleneck with many replicas and many attributes - need to optimize fast writes and reads with less updates
- Fixed schema is not sufficient … as application becomes popular more attributes need to be captured and DB modelling becomes an issue.
  - ✓ Which attributes are used depends on the use case.

replicas of shards in a social site DB




```
{
  "title": "Sweet fresh strawberry",
  "type": "fruit",
  "description": "Sweet fresh strawberry",
  "image": "1.jpg",
  "weight": 250,
  "expiry": 30/5/2021,
  "price": 29.45,
  "avg_rating": 4
  "reviews": [
     { "user" : "p1",  "rating": 2, "review": " ….. " }, …
  ]
}
```

want to add field applicable to some products

Issues with RDBMS (2)

- Cannot handle very wide denormalized attribute sets
- Data layout formats - column or row major - depends on use case
  - ✓ What if we query only few columns ? Do I need to touch the entire row in storage layer ?
- Expensive to retain and query long term data - need low cost solution

```
{
  "title": "Sweet fresh strawberry",
  "type": "fruit",
  "description": "Sweet fresh strawberry",
  “image": "1.jpg",
  “weight": 250,
  "expiry": 30/5/2021,
  "price": 29.45,
  “avg_rating": 4
  “reviews”: [
     { “user” : “p1”,  “rating”: 2, “review”: “ ….. “ }, …
  ]
}
```

what if a JSON had 1000+ attributes
demographic records
for millions of users

RDBMS Bottleneck in scalability

- CPU clock speed has hit the thermal barrier
- Processors moving to multi-core
- Multi-core adaptation needs software redesign (No free lunch)
- RDBMS engines are monolithic software systems
- Originally designed to run on single CPU systems
- Not core aware – not fully multi-threaded to take advantage of all cores
- For more performance, upgrade servers - Enjoy free performance lunch
- Upgrading a server is an exercise that requires application downtime
- Given the relatively unpredictable user growth rate of modern software systems, there is either over or under provisioning of resources

Evolution of In Memory Data Grids and NoSQL helps in overcoming the limitations of traditional RDBMS used in modern web applications

Road Blocks to RDBMS Scaling

- RDBMS technology is a forced fit for modern interactive software systems
- RDBMS is incredibly complex internally, and changes are difficult
- Vendors of RDBMS technology have little incentive to disrupt a technology generating billions of dollars for them annually
- It requires huge investments to address the RDBMS scaling issue and find out a viable solution
- Techniques used to extend the useful scope of RDBMS technology fight symptoms but not the disease itself

Provocative statement:

The relational database will be a footnote in history, because of fundamental flaws in the RDBMS approach to managing relational data of the present era

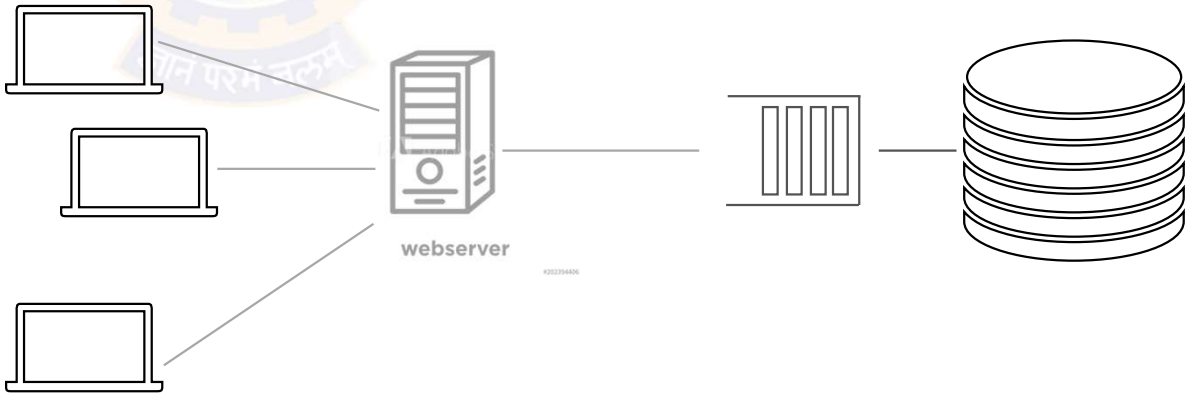
Scaling of RDBMS

- RDBMS can scale up (Vertical scaling) on a bigger server – Enjoys free performance lunch provided by Moore's law
- When the capacity of single server is reached, solution is to scale out and distribute the load across multiple servers
- RDBMS were designed for single CPU systems – scale out bottleneck
- This is when the complexity of relational databases starts to rub against their potential to scale
- Began to look at multi-node database solutions known as 'scaling out' or 'horizontal scaling'
- Different for approaches for RDBMS scaling include:
  - ✓ Queuing
  - ✓ Master-Slave
  - ✓ Sharding

Scaling with intermediate layer

Using a queue

- Portal is very popular, lot of users visiting it
  - ✓ Many users are concurrently visiting the pages of portal
  - ✓ Every time a page is visited, database needs to be updated to keep track of this visit
  - ✓ Database write is heavy operation
  - ✓ Database write is now a bottleneck !
- Solution
  - ✓ Use an intermediate queue between the web server and database
  - ✓ Queue will hold messages
  - ✓ Messages will be updated in the queue
  - ✓ Messages will not be lost



Scaling out RDBMS – Master/Slave

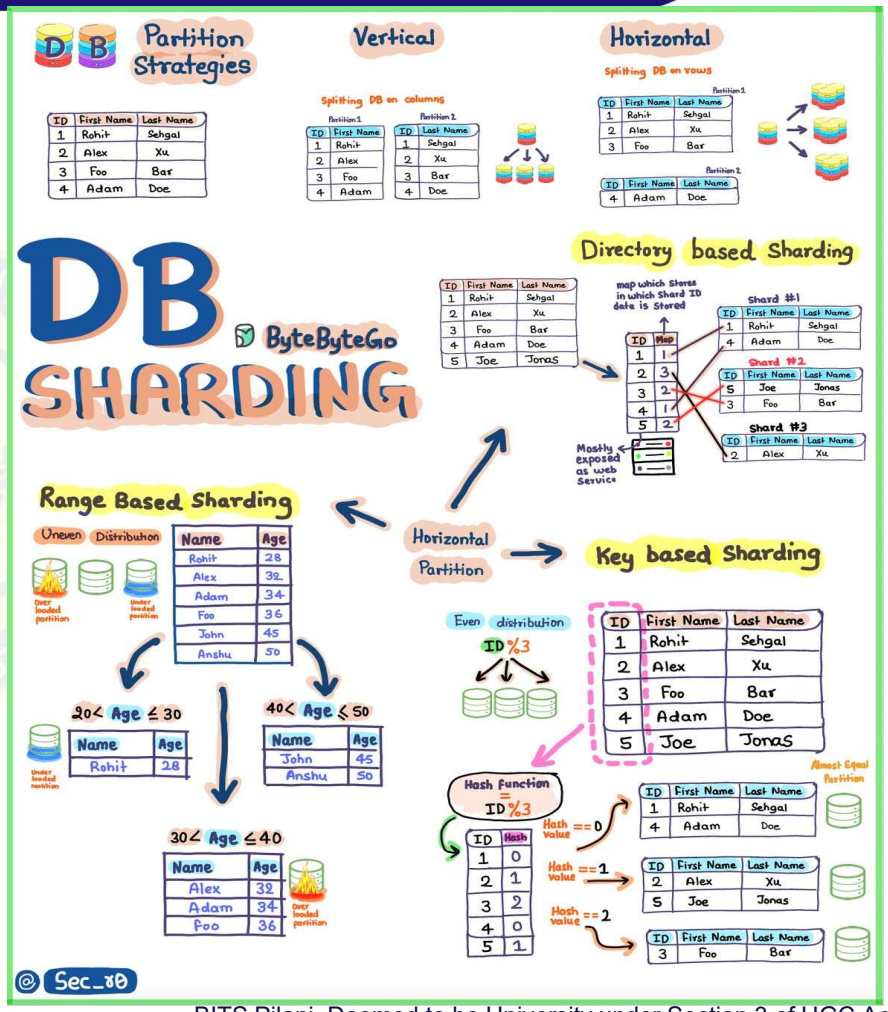
- All writes are written to the master.
- All reads performed against the replicated slave databases
- Good for mostly read, very few update applications
- Critical reads may be incorrect as writes may not have been propagated down
- Large data sets can pose problems as master needs to duplicate data to slaves

Scaling out RDBMS- Sharding

- The word *shard* means "a small part of a whole."
- Sharding is a type of database partitioning that separates large databases into smaller, faster, more easily managed parts.
- These smaller parts are called data shards and are hosted on multiple machines on same or different types of databases (MySQL/PostgreSQL)
- Need to manage parallel access in the application
- Scales well for both reads and writes
- **Not transparent, application needs to be partition-aware**

Scaling with Database Partitions (Sharding)

- Application is too popular
  - ✓ Users are using it very heavily, increasing the load on application
  - ✓ Maintaining the page view count is becoming difficult even with queue
- Solution
  - ✓ Use database partitions / shards
  - ✓ Data is divided into partitions (shards) which are hosted on multiple machines
  - ✓ Database writes are parallelized
  - ✓ Scalability increasing
  - ✓ Also, complexity is increasing!
- Sharding Strategies
  - ✓ Vertical
  - ✓ Horizontal
  - ✓ Directory based
  - ✓ Range based
  - ✓ Key based

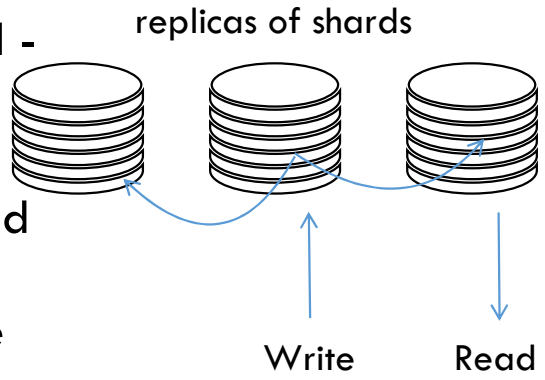


Difference between Sharding and RDBMS Partitioning

- Sharding and partitioning break up a large database into smaller databases
- But, there is a difference between the two methods.
  - After a database is sharded, the data in the new tables is spread across multiple systems
  - But, partitioning groups data subsets within a single database instance.

Issues with RDBMS sharding

- Too many shards results in data on too many disks - some disks are bound to fail
- Fault tolerance is needed for shard replicas - so more things to manage
- Complex logic to read / write because need to locate the right shard - human errors can be devastating
- Keep re-sharding and balancing as data grows or load increases
- What is the consistency semantics of updating replicas ? Should a read on a replica be allowed before it is updated ?
- Is it optimized when data is written once and read many times or vice versa ?

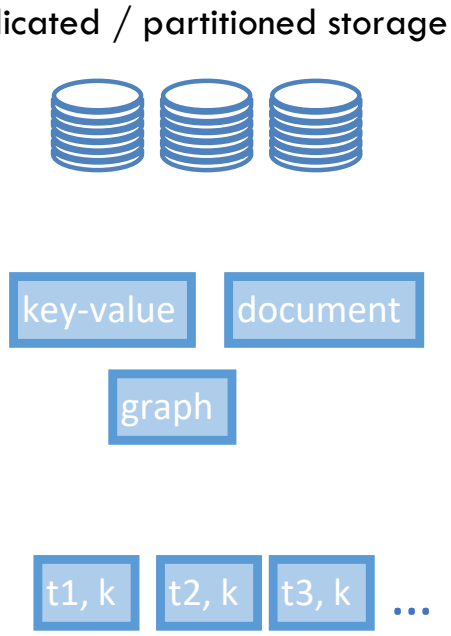


Topics for today

- Motivation
  - ✓ Why do modern Enterprises need to work with data
  - ✓ What is Big Data and data classification
  - ✓ Scaling RDBMS
- What is a Big Data System
  - ✓ Characteristics
  - ✓ Design challenges
- Architecture
  - ✓ High level architecture of Big Data solutions
  - ✓ Technology ecosystem
  - ✓ Case studies

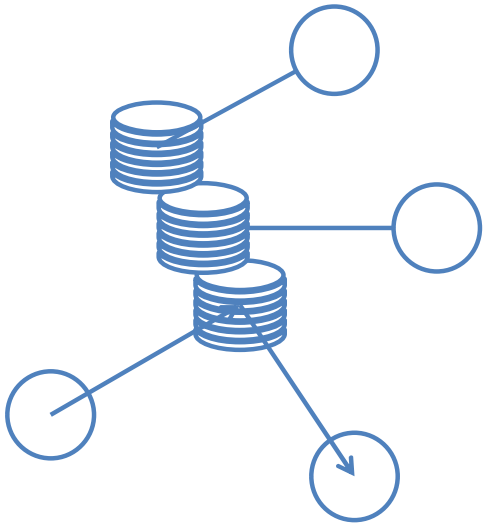
Desirable Characteristics of Big Data Systems (1)

- Application does not need to bother about common issues like sharding, replication ,etc
  - ✓ Developers more focused on application logic rather than data management
- Flexible schema for easy data modelling
  - ✓ Not necessary that every record has same set of attributes
- If possible, treat data as immutable
  - ✓ Keep adding timestamped versions of data values
  - ✓ Avoid human errors by not destroying an existing good copy
  - ✓ Handle high data volume, at very fast rate coming from variety of sources because immutable writes are faster



Desirable Characteristics of Big Data Systems (2)

- Application specific consistency models
  - ✓ a reader may read a replica that's has not been updated yet as in "read preference" options in MongoDB
  - ✓ e.g. comments on social media
- Tunable consistency for faster reads and writes



Cassandra is a NoSQL database for write-heavy workload and eventual consistency

Desirable Characteristics of Big Data Systems (3)

- Built as distributed and incrementally scalable systems
  - ✓ add new nodes to scale as in a Hadoop cluster
- Options to have cheaper long term data retention
  - ✓ long term data reads can have more latency and can be less expensive to store on commodity hardware, e.g. Hadoop file system (HDFS)
- Generalized programming models that work close to the data
  - ✓ e.g. Hadoop Map-Reduce that runs tasks on data nodes

Challenges in Big Data Systems (1)

- Latency issues in algorithms and data storage working with large data sets
- Reliability, availability, consistency - Basic design considerations of Distributed and Parallel systems
- What data to keep and for how long - depends on analysis use case
- Cleaning / Curation of data
- Overall orchestration involving large volumes of data
- Choose the right technologies from many options, including open source, to build the Big Data System for the use cases

https://www.sqltutorial.org/

Topics for today

- Motivation
  - ✓ Why do modern Enterprises need to work with data
  - ✓ What is Big Data and data classification
  - ✓ Scaling RDBMS
- What is a Big Data System
  - ✓ Characteristics
  - ✓ Design challenges
- Architecture
  - ✓ High level architecture of Big Data solutions
  - ✓ Technology ecosystem
  - ✓ Case studies
- Summary

Needs of Big Data Systems

- Processing of large data volume
- CPU intensive and prolonged computations
- Scalability enables increase or decrease in the capacity of data storage and processing, as per the complexity of computations and volume of data
- Types of Scalability
  - ✓ Vertical
  - ✓ Horizontal
  - ✓ Elastic

Vertical Scalability (Scaling Up)

- Scaling up the given system's resources and increasing the system's analytics, reporting and visualization capabilities
- Solve problems of greater complexities by scaling up
- For example, *x* TB of data take time *t* for processing. When code size with increasing complexity increase by factor *n*, then scaling up means that processing takes equal, less or much less than ($n \times t$) for *x* TB.
- How vertical scalability is achieved
  - ✓ Server changes / upgradations
    - More powerful CPU
    - More memory
    - Product companies
      - Enjoy Free Performance Lunch facilitated by Moore's law
      - Exploited by RDBMS aka SQL Databases by new releases

Horizontal Scalability (Scaling Out)

- Horizontal scalability means increasing the number of systems working in coherence and scaling out the workload
- Processing different datasets of a large dataset by increasing number of systems running in parallel.
- Scaling out means using more resources and distributing the processing and storage tasks in parallel
- If *r* resources in a system process *x* TB of data in time *t*, then the ($p_x x$) TB on *p* parallel distributed nodes such that the time taken up remains *t* or is slightly more than *t*
- How Horizontal scalability is achieved:
  - ✓ Parallelization of jobs at several levels:
    - (i) Distributing separate tasks onto separate threads on the same CPU,
    - (ii) Distributing separate tasks onto separate CPUs on the same computer and
    - (iii) Distributing separate tasks onto separate computers

Elastic Scaling – Cloud computing

Cloud computing

- on-demand service
- resource pooling,
- scalability,
- accountability, and
- broad network access.

Elastic scaling (scaling up and scaling down) by dynamic provisioning based on computational need

Types of Big Data solutions

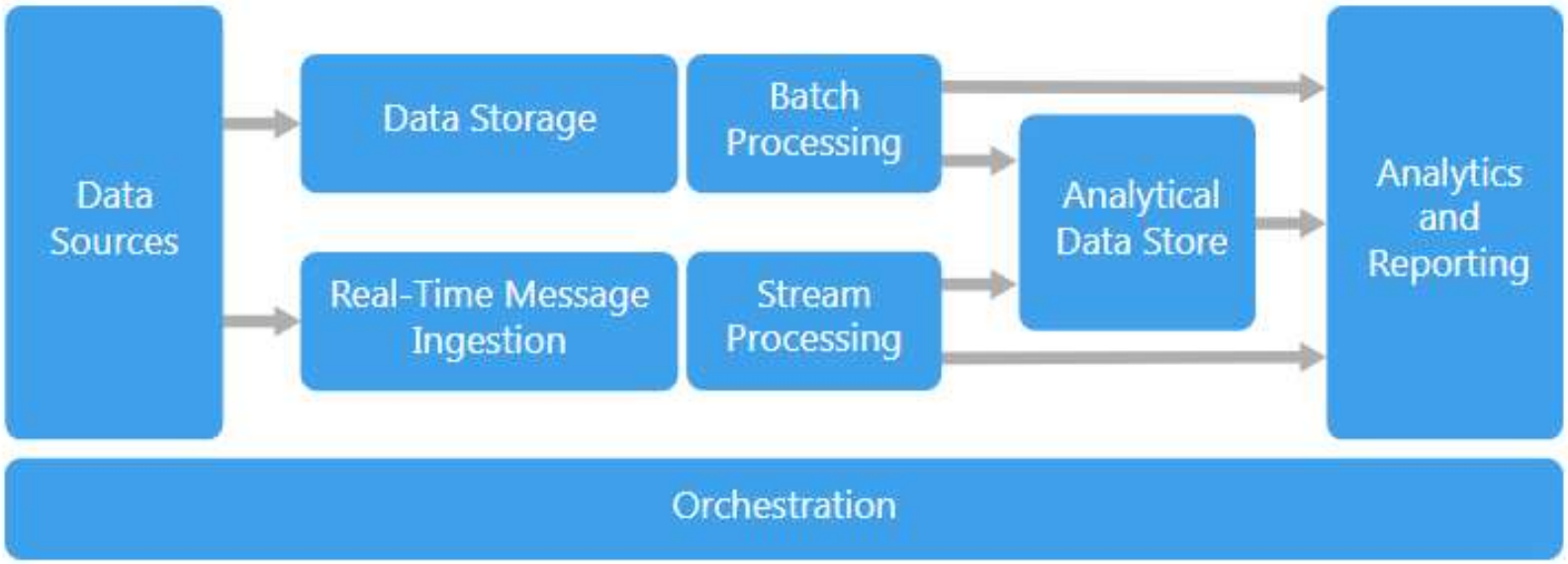
1. Batch processing of big data sources at rest
   - ✓ Building ML models, statistical aggregates
   - ✓ "What percentage of users in US last year watched shows starring Kevin Spacey and completed a season within 4 weeks or a movie within 4 hours"
   - ✓ "Predict number of US family users in age 30-40 who will buy a Kelloggs cereal if they purchase milk"

2. Real-time processing of big data in motion
   - ✓ Fraud detection from real-time financial transaction data
   - ✓ Detect fake news on social media platforms

3. Interactive exploration with ad-hoc queries
   - ✓ Which region and product has least sales growth in last quarter

Big data architecture style

- Designed to handle the ingestion, processing, and analysis of data that is too large or complex for traditional database systems.



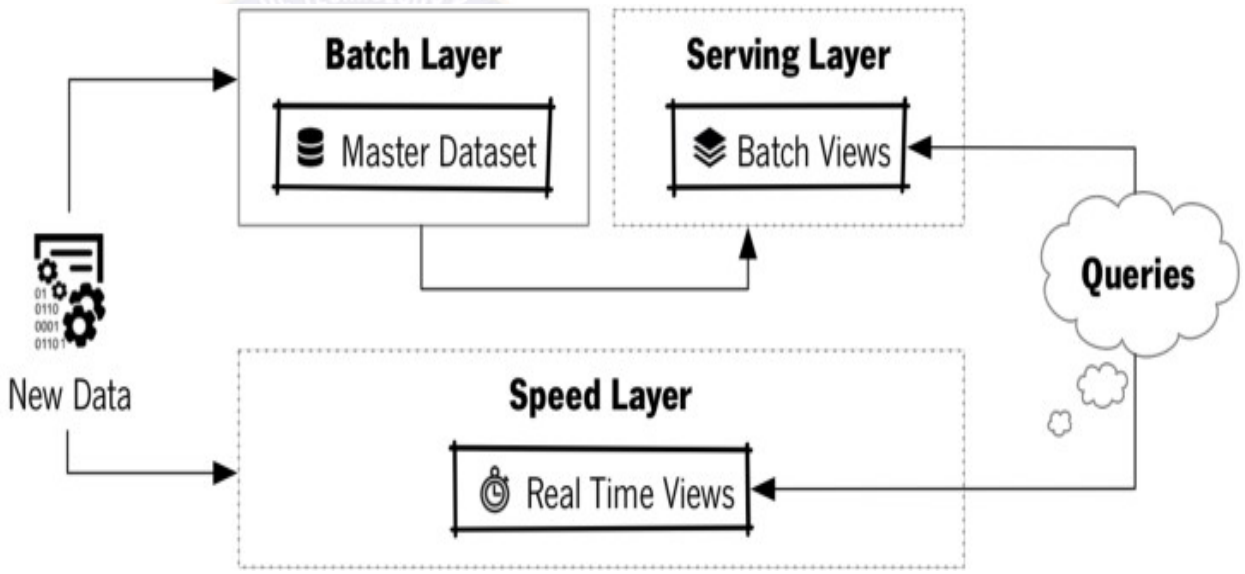
Source : Microsoft Big Data Architecture

Kappa Architecture: Stream Processing in Big Data Analytics

Lambda Architecture

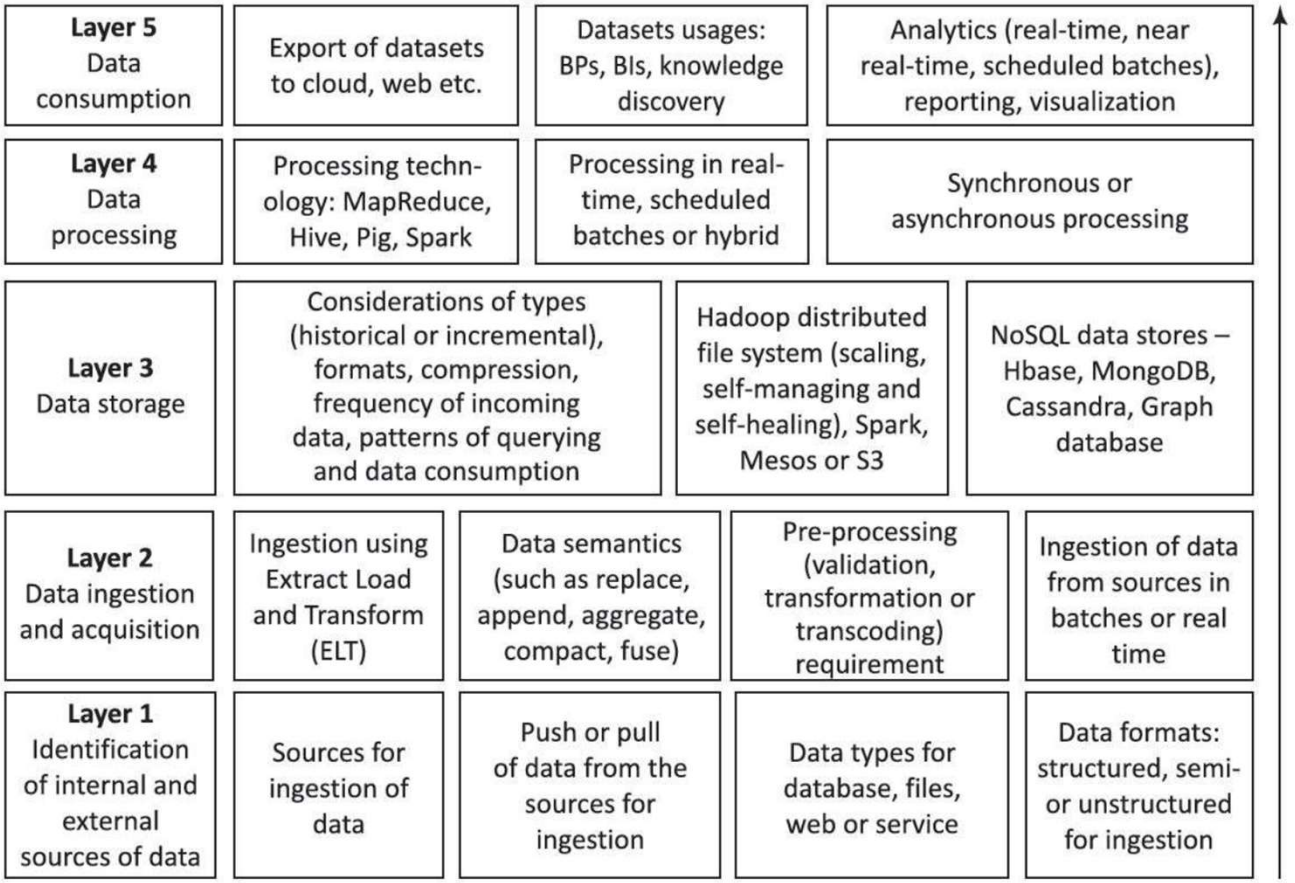
- Lambda architecture is a way of processing massive quantities of data (i.e. “Big Data”) that provides access to batch-processing and stream-processing methods with a hybrid approach .
- Lambda architecture is used to solve the problem of computing arbitrary functions in real time.
- The lambda architecture is composed of 3 layers:

1. Batch Layer
2. Serving Layer
3. Speed Layer (Stream Layer)



What Is Lambda Architecture? (databricks.com)
Source: http://vda-lab.github.io/2019/10/lambda-architecture

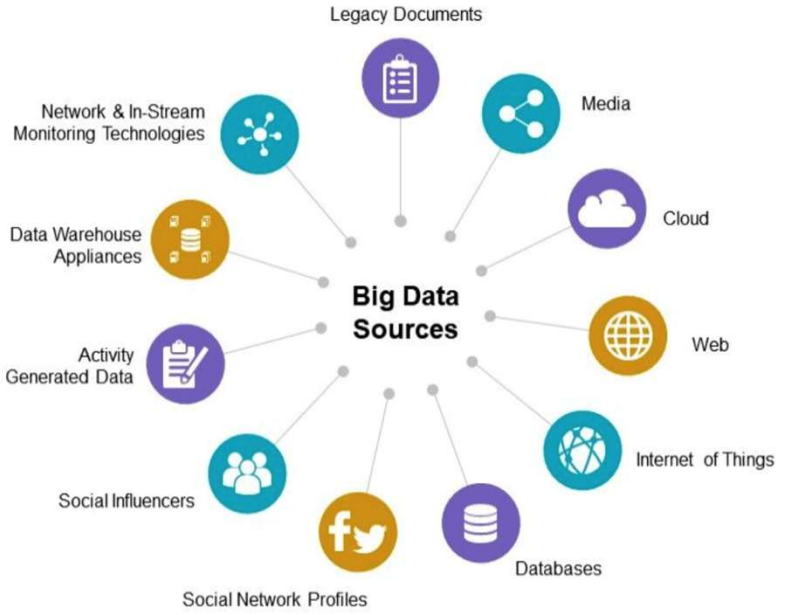
Logical Layers in Big Data Processing Architecture



Big Data Systems Components (1)

1. Data sources
   - ✓ One or more data sources like databases, docs, files, IoT devices, images, video etc.

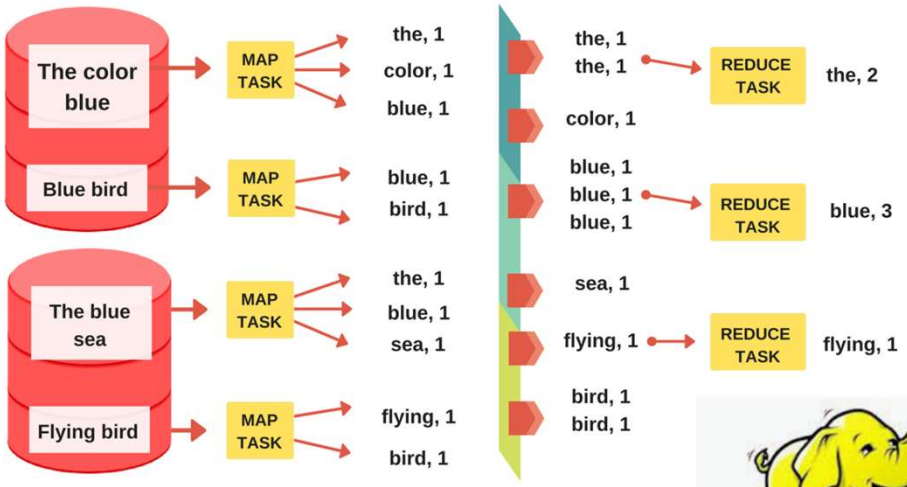
2. Data Storage
   - ✓ Data for batch processing operations is typically stored in a distributed file store that can hold high volumes of large files in various formats.
   - ✓ Data can also be stored in key-value stores.



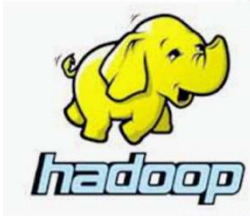
e.g. social data

e.g. medical images

Big Data Systems Components (2)



e.g. search scans on unindexed docs



3. Batch processing
   - ✓ Process data files using long-running parallel batch jobs to filter, sort, aggregate or prepare the data for analysis.
   - ✓ Usually, these jobs involve reading source files, processing them, and writing the output to new files.

4. Real-time message ingestion
   - ✓ Capture data from real-time sources and integrate with stream processing. Typically, these are in-memory systems with optional storage backup for resiliency.

Big Data Systems Components (3)

5. Stream processing

Real-time in-memory filtering, aggregating or preparing the data for further analysis. The processed stream data is then written to an output sink. These are mainly in-memory systems. Data can be written to files, database, or integrated with an API.

e.g. fraud detection logic

6. Analytical data store

Real-time or batch processing can be used to prepare the data for further analysis. The processed data in stored in a structured format to be queried using analytical tools. The analytical data store used to serve these queries can be a Kimball-style relational data warehouse or BigData warehouse like Hive. There may be also NoSQL stores such as MongoDB, HBase.

e.g. financial transaction history across clients for spend analysis

Big Data Systems Components (4)

e.g. weekly management report

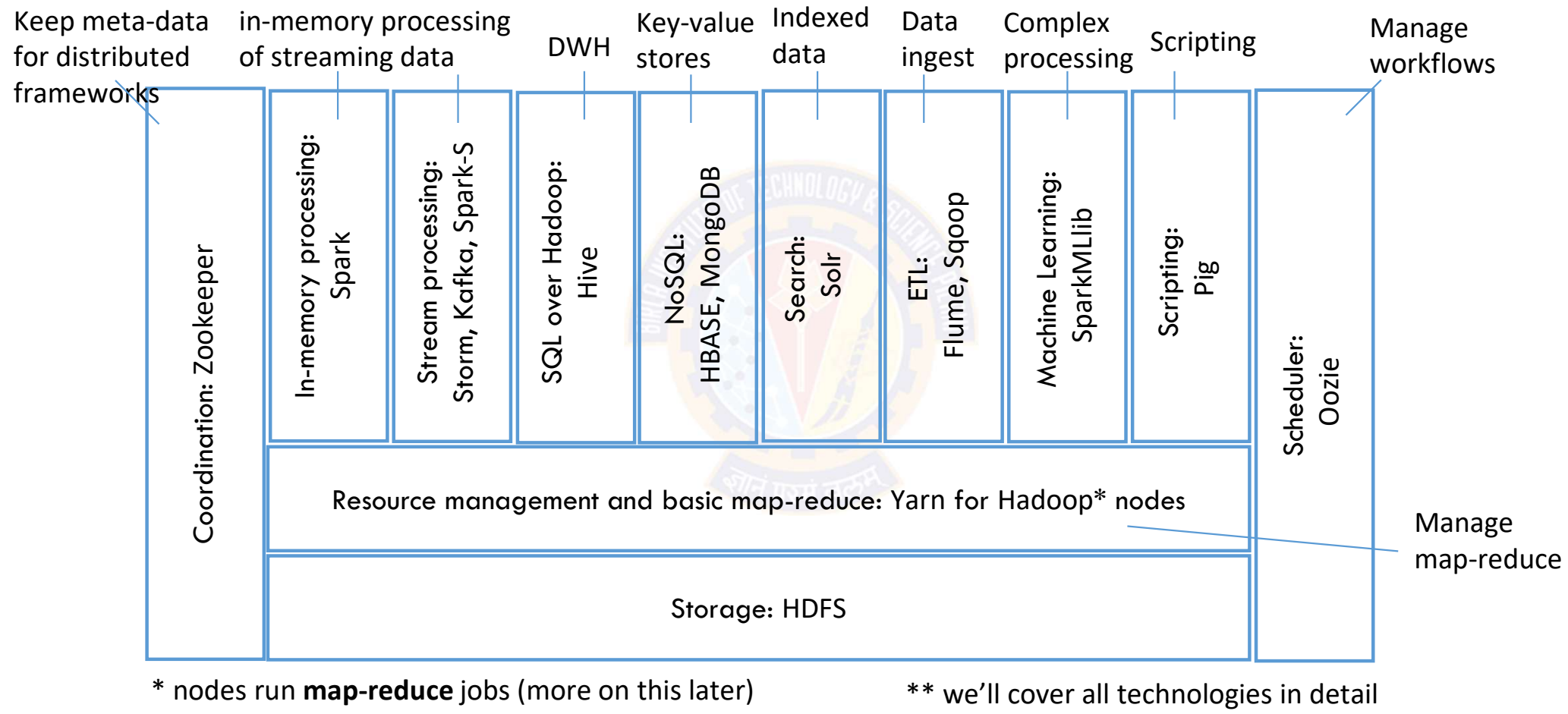
7. Analysis and reporting

The goal of most big data solutions is to provide insights into the data through analysis and reporting. These can be various OLAP, search and reporting tools.

8. Orchestration and ETL

Most big data solutions consist of repeated data processing operations, encapsulated in workflows, that transform source data, move data between multiple sources and sinks, load the processed data into an analytical data store, or push the results straight to a report or dashboard. To automate these workflows, you can use an orchestration technology such Azure Data Factory, Apache Oozie or Sqoop.

Technology Ecosystem (showing mostly Apache projects)



* nodes run **map-reduce** jobs (more on this later)

** we'll cover all technologies in detail

Case Study: IT Ops

Using Big Data tools and architecture for managing IT

IT Operations Analytics

- IT systems generate large volumes of monitoring, logging and event data
- Can we use this data to proactively look for anomalous patterns and predict an issue
- Can we localise possible root causes
- Can we help an engineer quickly explore the data to confirm the specific root cause

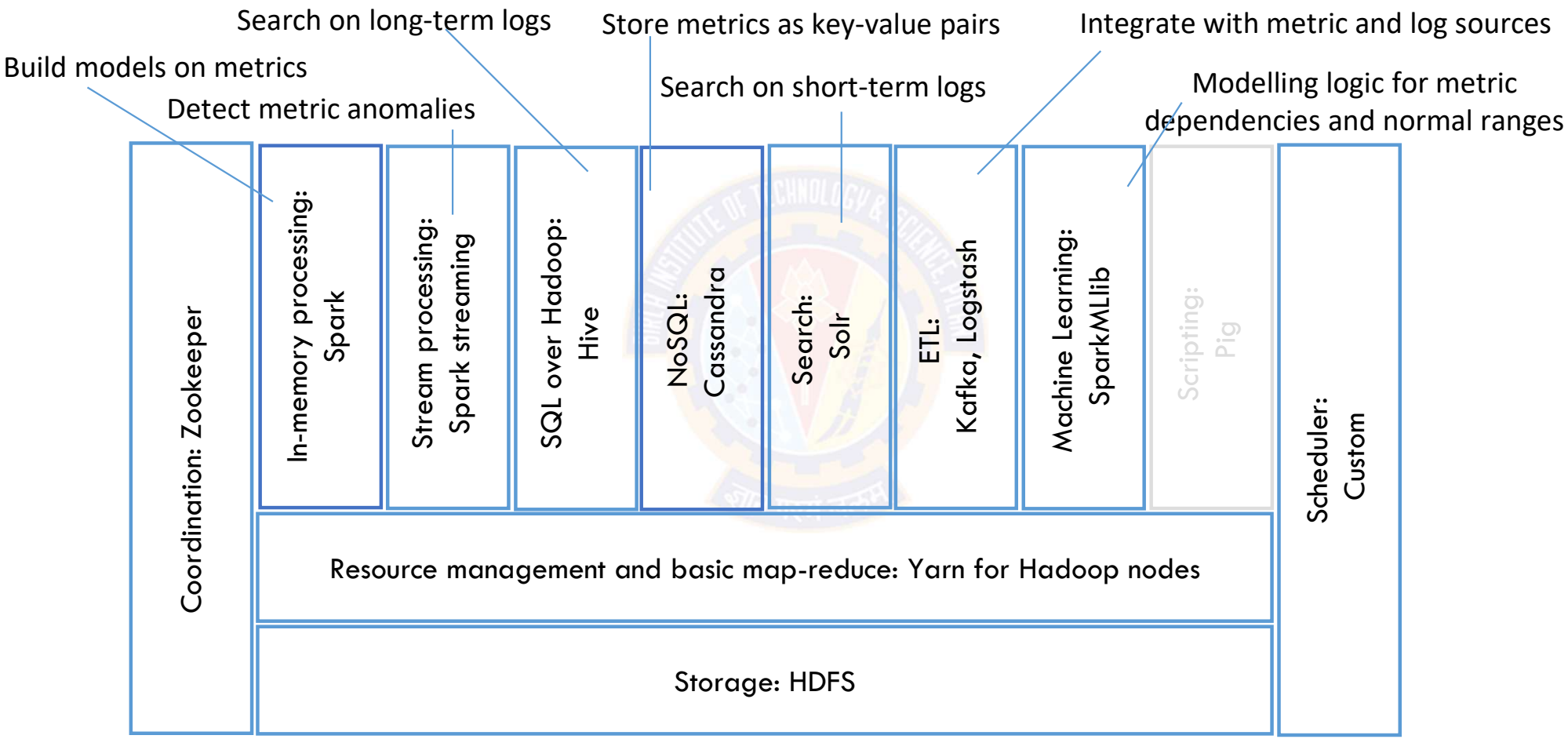
IT Operations Analytics

- IT systems generate large volumes of monitoring, logging and event data
- Can we use this data to proactively look for anomalous patterns and predict an issue
- Can we localize possible symptoms and possible causes
- Can we help an engineer quickly explore the data to confirm the specific root cause

Real-time streaming analysis of metrics
- in-memory fast compute
- real-time model updates and model lookups

Interactive time-sensitive search and exploration of log and metric data
- older data may take time (has this happened earlier in last month)
- but new data should be fast (what happened few minutes back)

Big Data Platform



Where will you apply this architecture style

- Consider this architecture style when you need to:
  - ✓ Store and process data in volumes too large for a traditional database
  - ✓ Handle semi-structured or data with evolving structure - e.g. demographic data with hundreds of attributes
  - ✓ Transform unstructured data for analysis and reporting
  - ✓ Capture, process, and analyze **unbounded streams of data** with low latency
  - ✓ Capture, process, and analyze **unbounded historical data** cost effectively

Big data architecture benefits

- Technology choices
  - ✓ Variety of technology options in open source and from vendors are available
- Performance through parallelism
  - ✓ Big data solutions take advantage of data or task parallelism, enabling high-performance solutions that scale to large volumes of data.
- Elastic scale
  - ✓ All of the components in the big data architecture support scale-out provisioning, so that you can adjust your solution to small or large workloads and pay only for the resources that you use.
- Flexibility with consistency semantics (more in CAP theorem)
  - ✓ E.g. Cassandra or MongoDB can make inconsistent reads for better scale and fault tolerance
- Good cost performance ratio
  - ✓ Ability to reduce cost at the expense of performance. E.g. long term data storage in commodity HDFS nodes.
- Interoperability with existing solutions
  - ✓ The components of the big data architecture are also used for IoT processing and enterprise BI solutions, enabling you to create an integrated solution across data workloads. e.g. Hadoop can work with data in Amazon S3.

Big data architecture challenges

- Complexity
  - ✓ Big data solutions can be extremely complex, with numerous components to handle data ingestion from multiple data sources. It can be challenging to build, test, and troubleshoot big data processes.
- Skillset
  - ✓ Many big data technologies are highly specialized, and use frameworks and languages that are not typical of more general application architectures. On the other hand, big data technologies are evolving new APIs that build on more established languages.
- Technology maturity
  - ✓ Many of the technologies used in big data are evolving. While core Hadoop technologies such as Hive and Pig have stabilized, emerging technologies such as Spark introduce extensive changes and enhancements with each new release.

Topics for today

- Motivation
  - ✓Why do modern Enterprises need to work with data
  - ✓What is Big Data and data classification
  - ✓Scaling RDBMS
- What is a Big Data System
  - ✓Characteristics
  - ✓Design challenges
- Architecture
  - ✓High level architecture of Big Data solutions
  - ✓Technology ecosystem
  - ✓Case studies

Summary

- Why modern Enterprises and new age applications are data-centric
- Challenges with existing data management systems
- Advantages and challenges with Big Data systems
- High level architecture and technology ecosystem
- Some real applications using the tech stack

Next Session:
Locality of Reference (LOR)