



BITS Pilani

Pilani|Dubai|Goa|Hyderabad

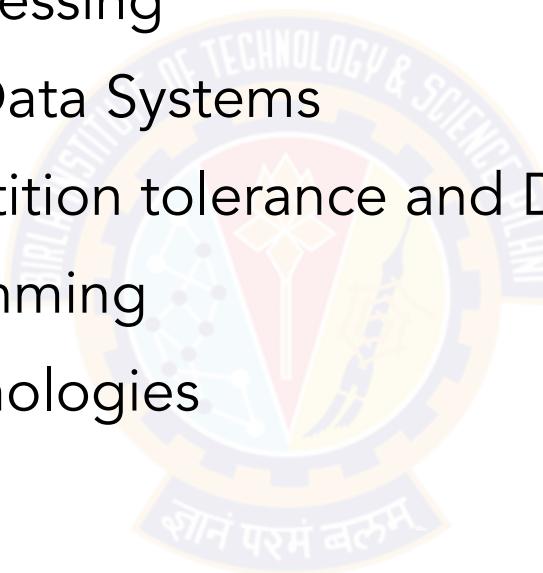
DSECL ZG 522: Big Data Systems

Session 1: Introduction to Big Data, Locality of Reference

Dr. Anindya Neogi,
Associate Professor (Off Campus)
anindya.neogi@pilani.bits-pilani.ac.in

Course Outline

- S1: Introduction to Big Data and data locality
- S2: Parallel and Distributed Processing
- S3: Big Data Analytics and Big Data Systems
- S4: Consistency, Availability, Partition tolerance and Data Lifecycle
- S5: Distributed Systems Programming
- S6-S9: Hadoop ecosystem technologies
- S10: NoSQL Databases
- S11: Big Data on Cloud
- S12: Amazon storage services
- S13-16: In-memory and streaming - Spark

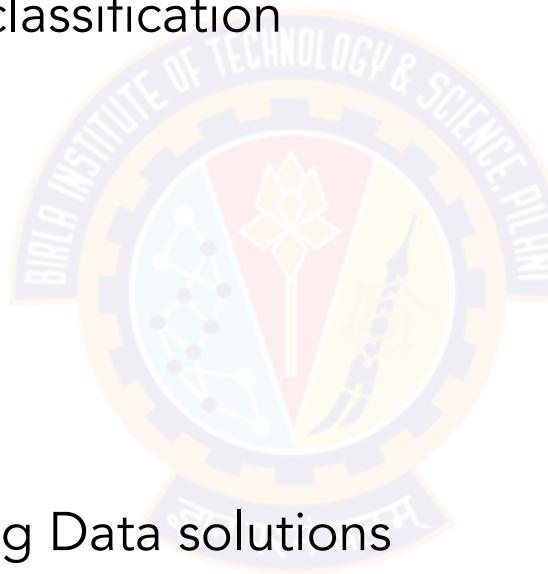


Books

T1	Seema Acharya and Subhashini Chellappan. <i>Big Data and Analytics</i> . Wiley India Pvt. Ltd. Second Edition
R1	DT Editorial Services. <i>Big Data - Black Book</i> . DreamTech. Press. 2016
R2	Kai Hwang, Jack Dongarra, and Geoffrey C. Fox. <i>Distributed and Cloud Computing: From Parallel Processing to the Internet of Things</i> . Morgan Kauffman 2011
AR	Additional Reading (As per topic)

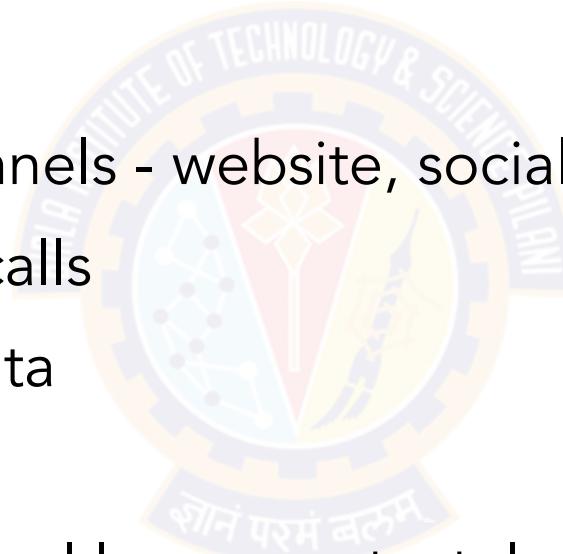
Topics for today

- 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
- Why locality of reference and storage organisation matter



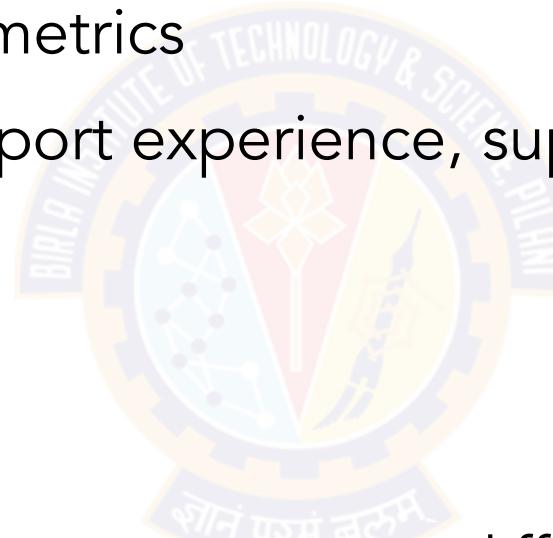
Example of a data-driven Enterprise: A large online retailer (1)

- 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.

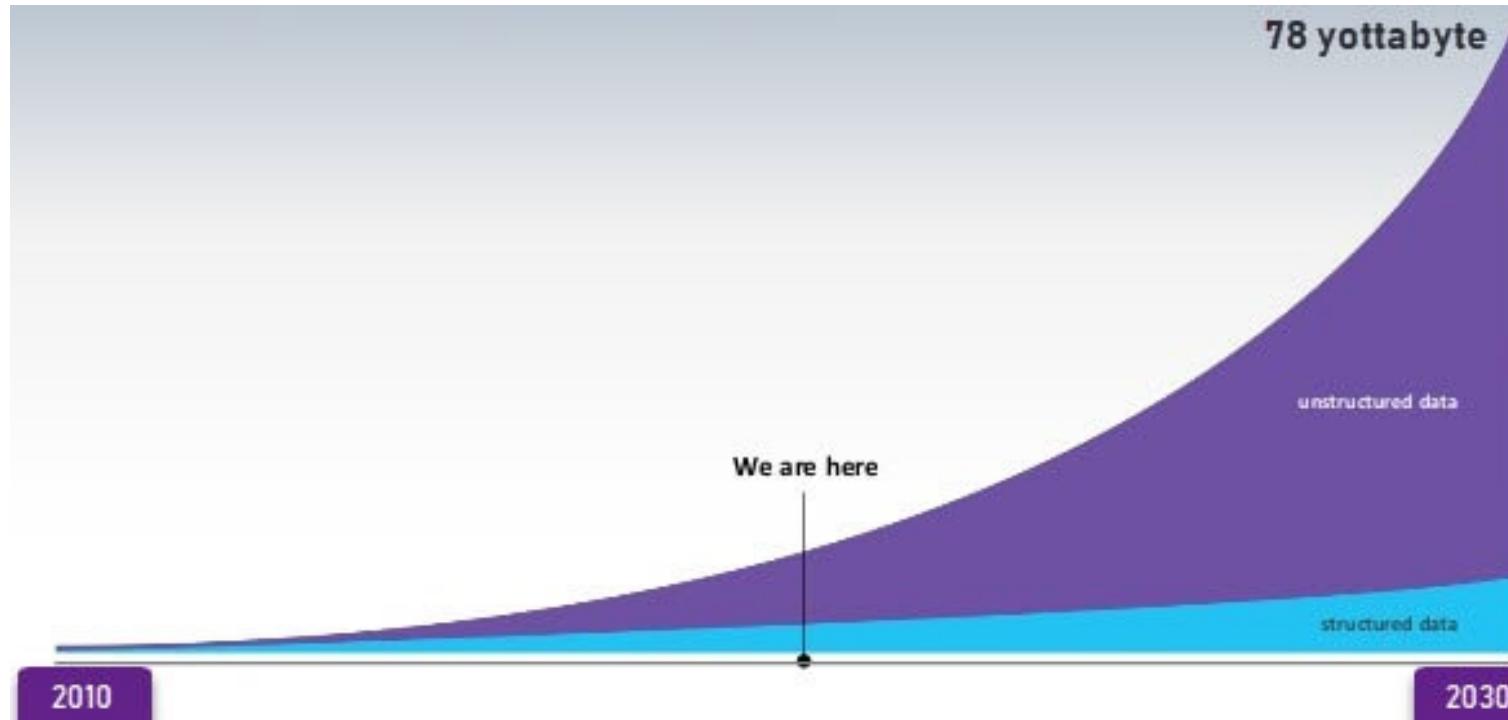


Example of a data-driven Enterprise: A large online retailer (2)

- 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



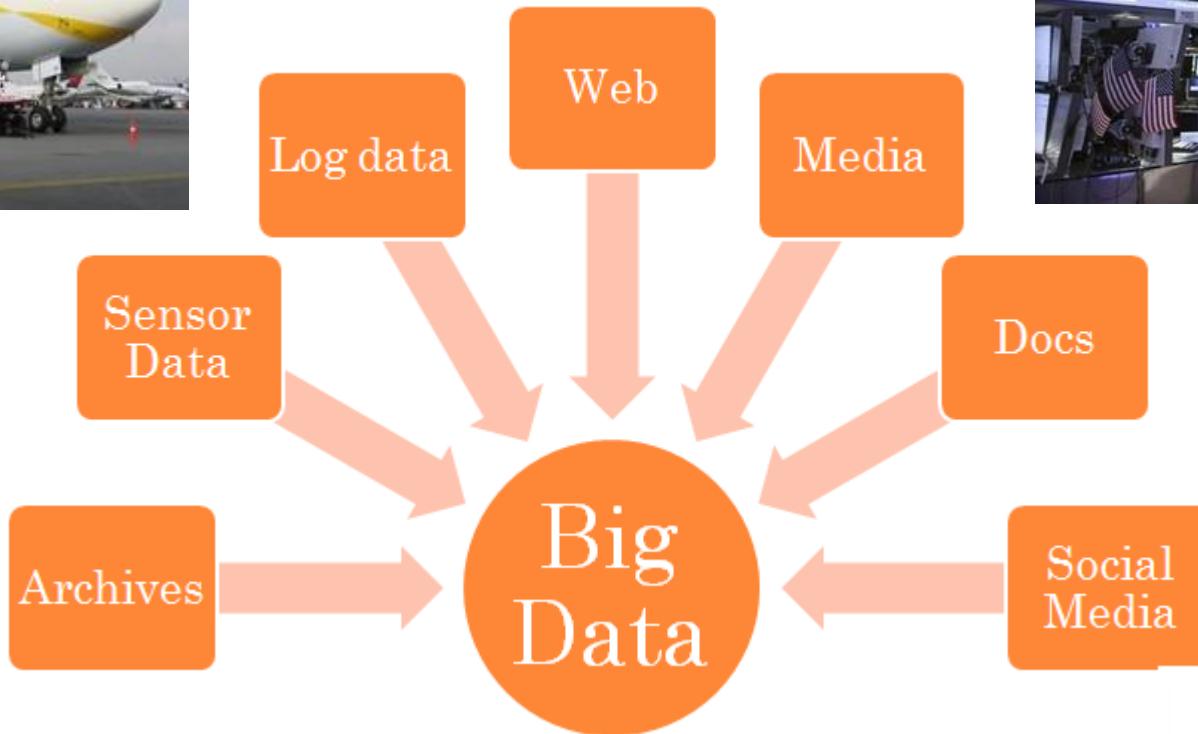
Data volume growth



- 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?](#)

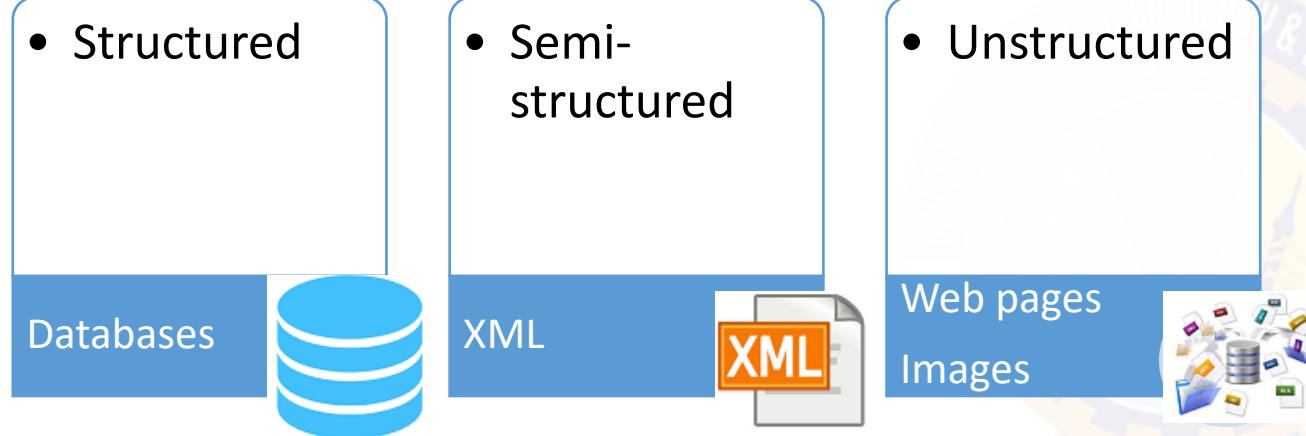
Variety of data sources



facebook

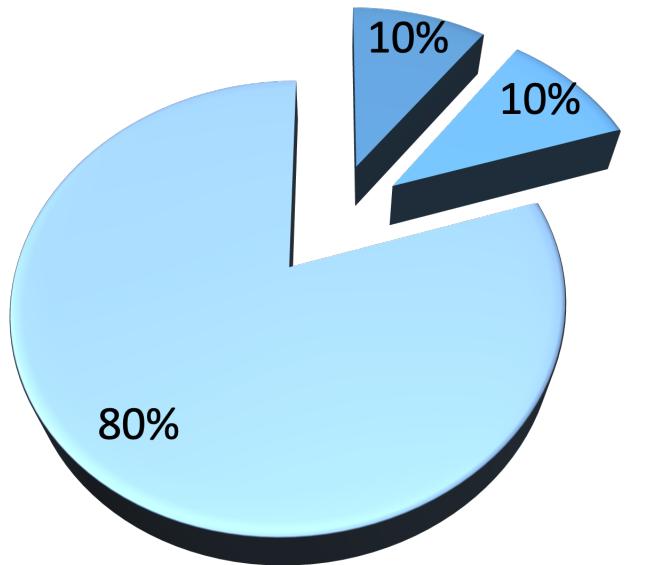
Source : What is Big Data?

Data classification

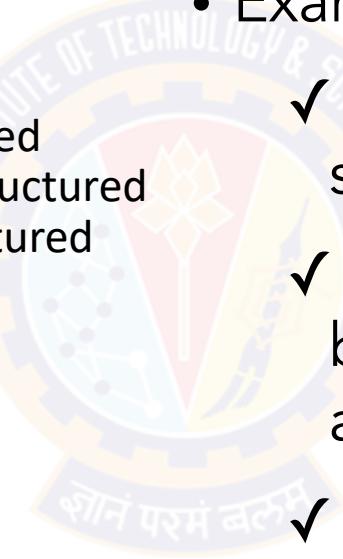


- Structured data is metrics, events that can be put in RDBMS with fixed schema
- Semi-structured data are XML, JSON structure where traditional RDBMS have support with varying efficiency but needs new kind of NoSQL databases
- New applications produce unstructured data which could be natural language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

Data usage pattern



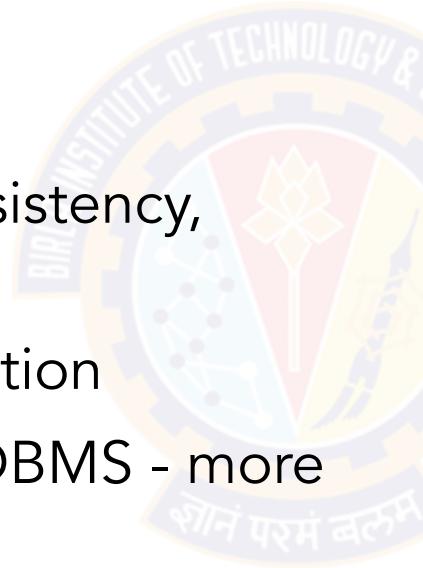
- Structured
- Semi-structured
- Unstructured



- Higher demand now of analysing 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 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

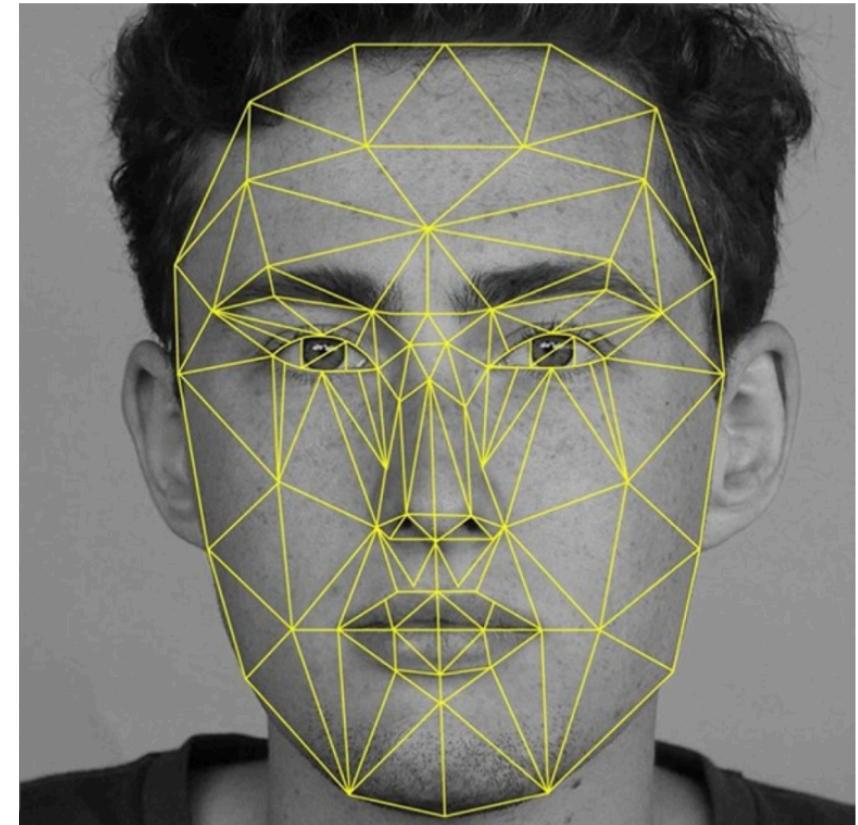
- No explicit data and schema separation
- Models real life situations better because attributes for every record could be different
- Easy to add new attributes
- XML, JSON structures
- Databases typically support flexible ACID properties, esp consistency of replicas
- Typically used by Systems of Engagement,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

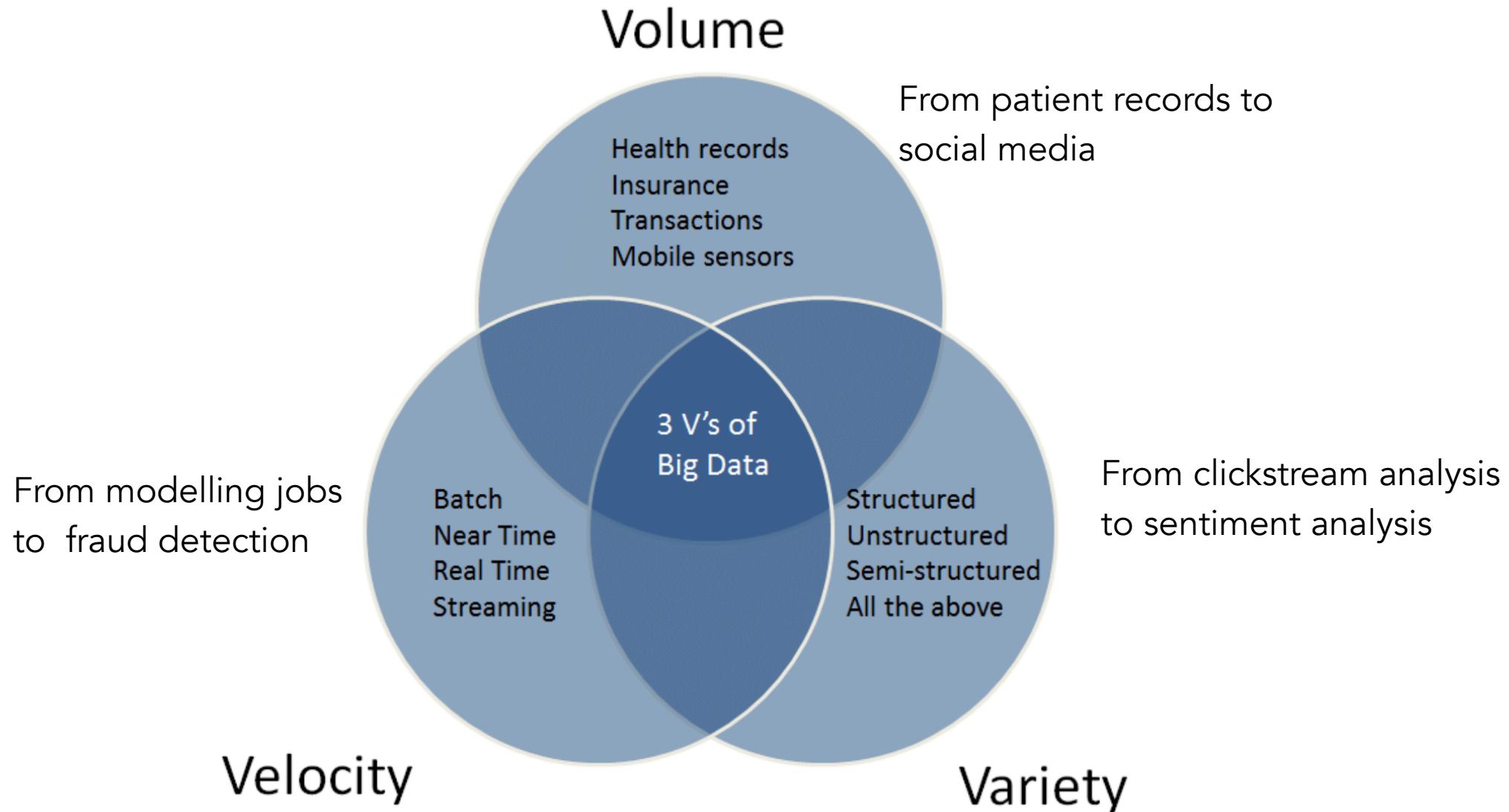
- More real-life data
 - ✓ video, voice, text, emails, chats, comments, reviews, blogs ...
- There is some structure that is typically extracted from the data depending on the use case
 - ✓ image adjustments at pixel structure level
 - ✓ face recognition from video
 - ✓ tagging of features extracted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

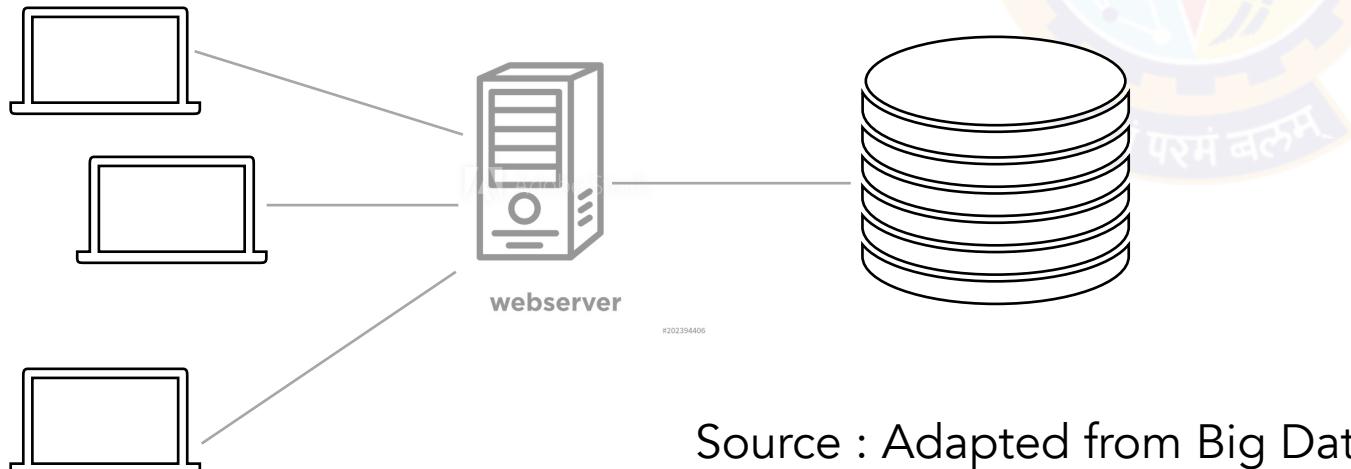
Define Big Data



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



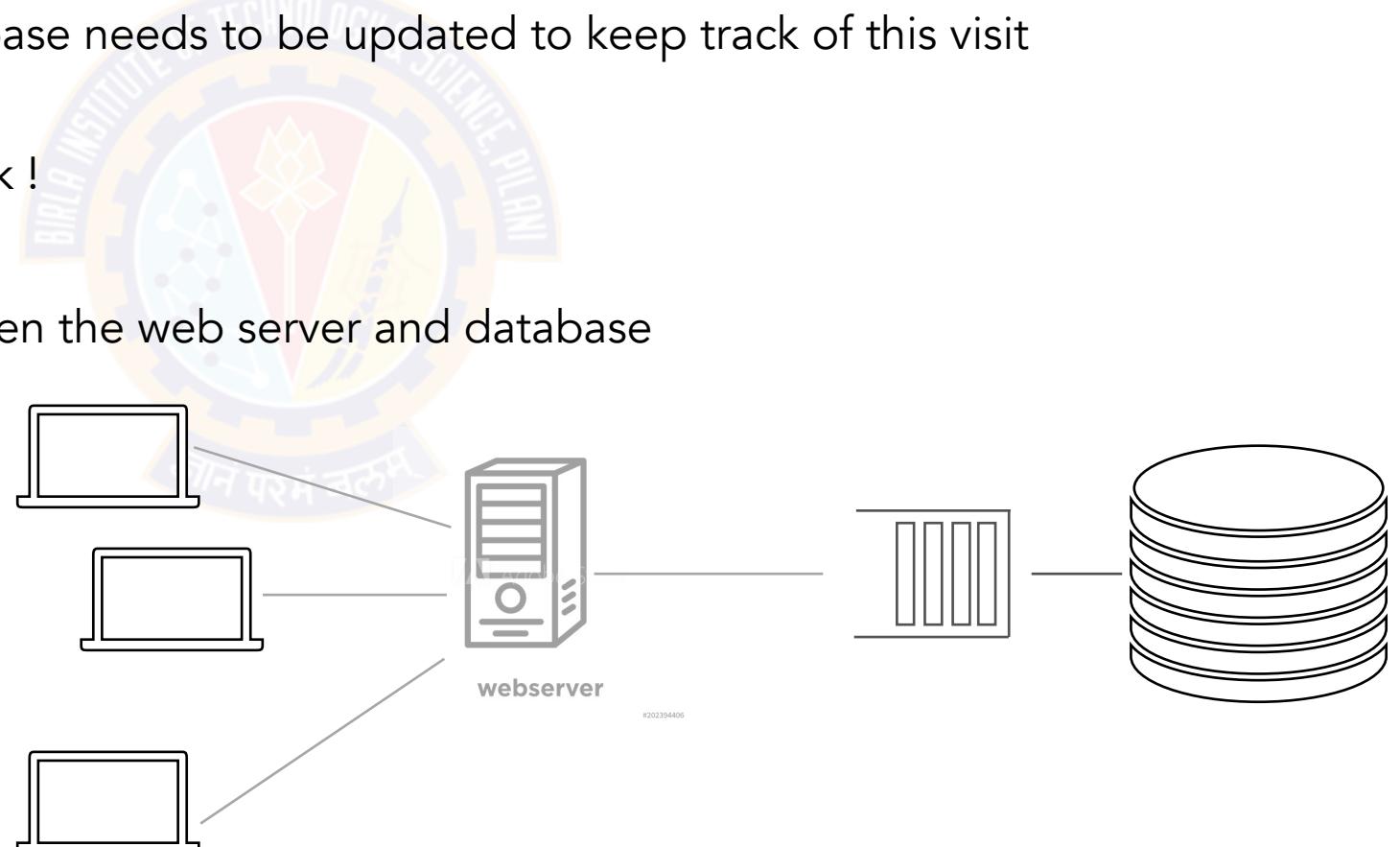
Column	Data Type
Id	Integer
User_ID	Integer
Page_URL	Varchar
Page_count	Long

Source : Adapted from Big Data by Nathan Marz

Scaling with intermediate layer

Using a queue

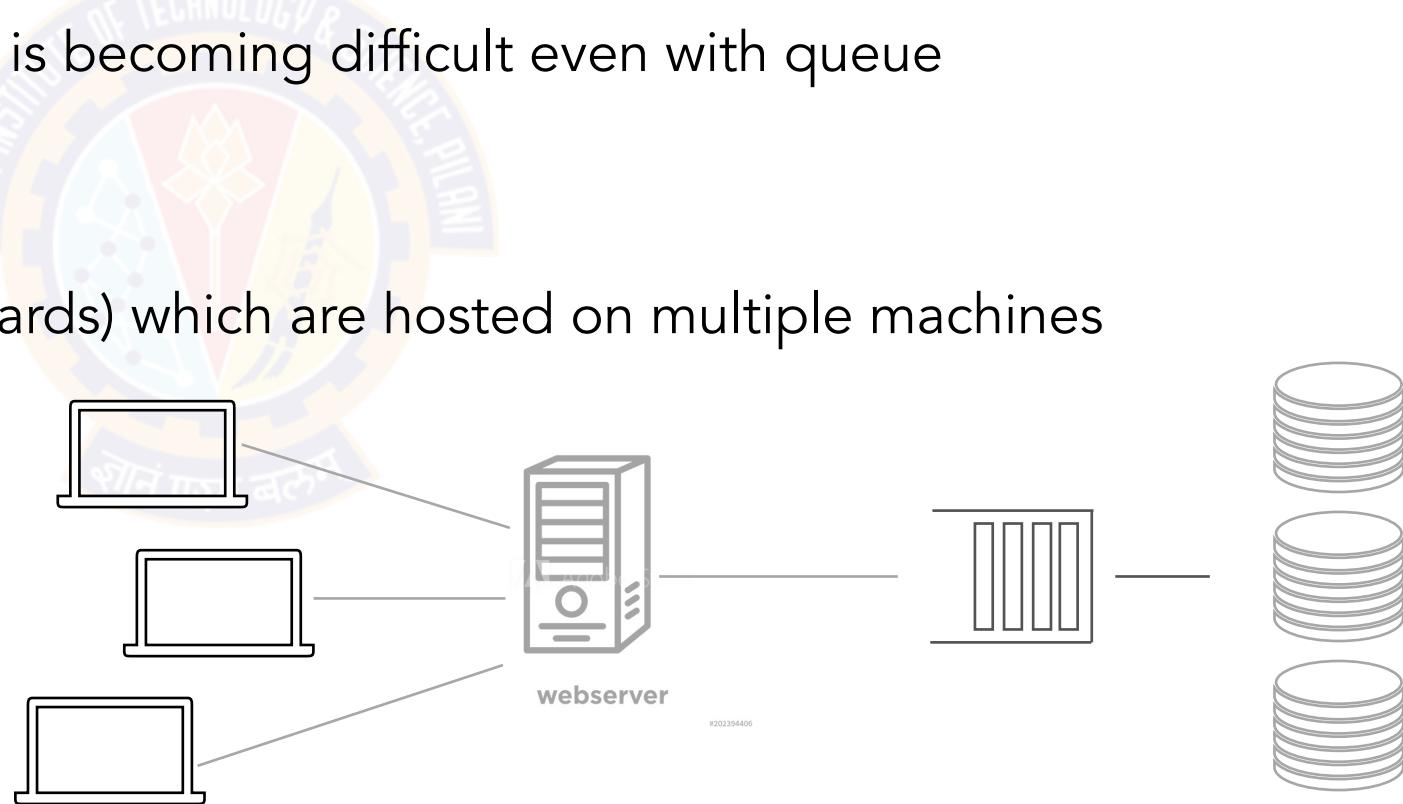
- Portal is very popular, lot of users visiting it
 - ✓ Many users are concurrently visiting the pages of portal
 - ✓ Every time a page is visited, database needs to be updated to keep track of this visit
 - ✓ Database write is heavy operation
 - ✓ Database write is now a bottleneck !
- Solution
 - ✓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



Scaling with Database Partitions

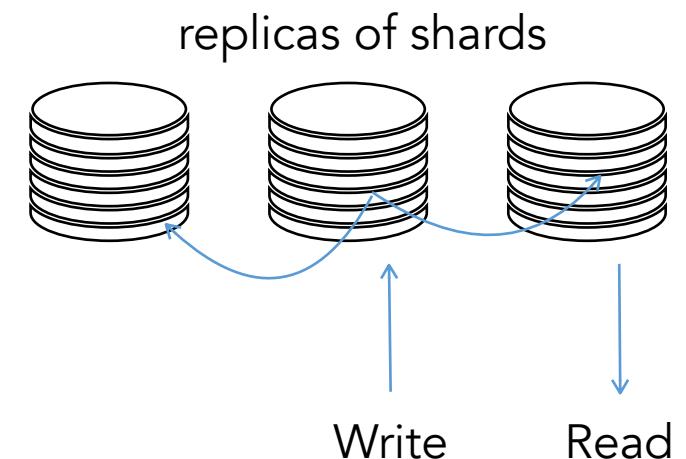
Using Database shards

- 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
 - ✓ Data is divided into partitions (shards) which are hosted on multiple machines
 - ✓ Database writes are parallelized
 - ✓ Scalability increasing
 - ✓ Also complexity increasing!



Issues with RDBMS (1)

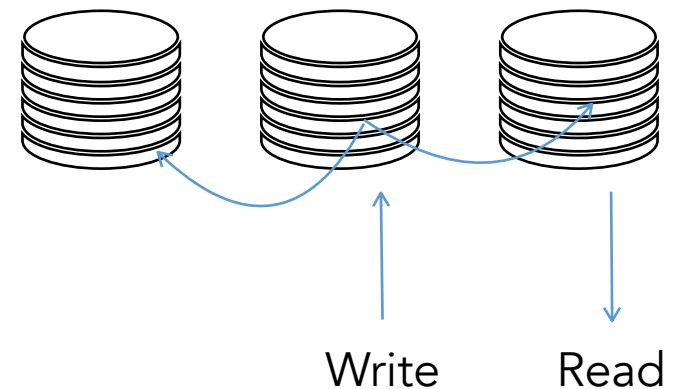
- With too many shards some disk is bound to fail
- Fault tolerance needs shard replicas - so more things to manage
- Complex logic to read / write because need to locate the right shard - human errors can be devastating
- Keep re-sharding and balancing as data grows or load increases
- What is the consistency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



Issues with RDBMS (2)

- Not all BigData use cases need strong ACID semantics, esp Systems of Engagement
 - ✓ Becomes a bottleneck with many replicas and many attributes - need to optimise 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

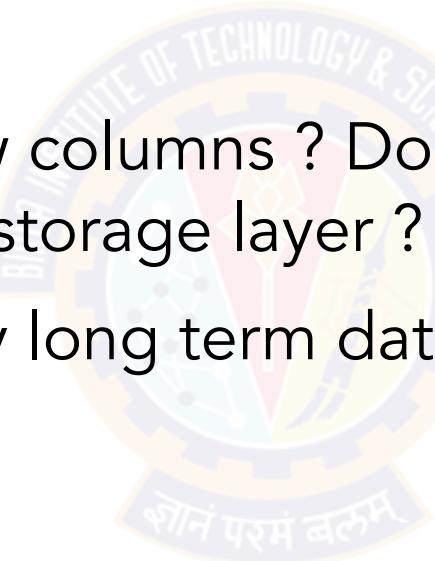


```
{  
    "title": "Sweet fresh strawberry",  
    "type": "fruit",  
    "description": "Sweet fresh strawberry",  
    "image": "1.jpg",  
    "weight": 250,  
    "expiry": 30/5/2021,  
    "price": 29.45,  
    "avg_rating": 4  
    "reviews": [  
        {"user": "p1", "rating": 2, "review": "....."},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

want to add field applicable to some products

Issues with RDBMS (3)

- Very wide de-normalized 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

Topics for today

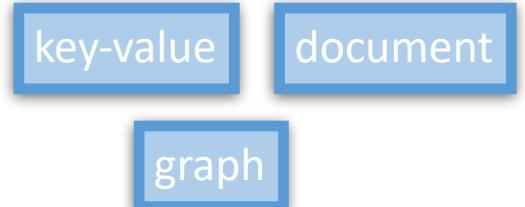
- Motivation
 - ✓ Why do modern Enterprises need to work with data
 - ✓ What is Big Data and data classification
 - ✓ Scaling RDBMS
- What is a Big Data System
 - ✓ Characteristics
 - ✓ Design challenges
- Architecture
 - ✓ High level architecture of Big Data solutions
 - ✓ Technology ecosystem
 - ✓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



Characteristics of Big Data Systems (1)

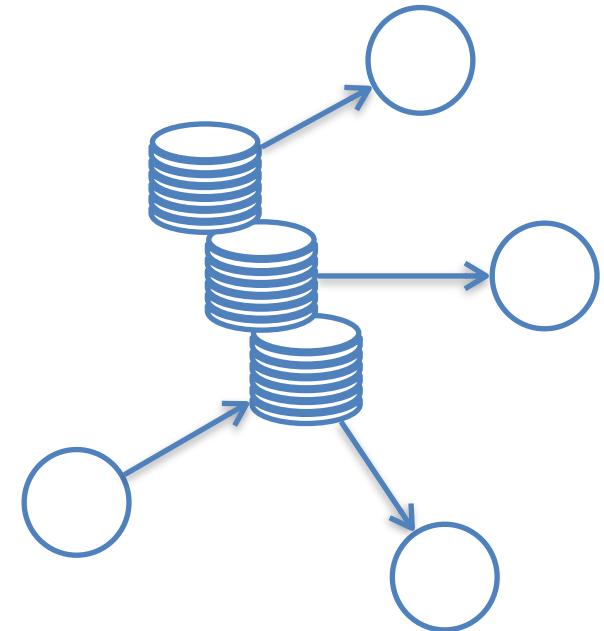
- Application does not need to bother about common issues like sharding, replication
 - ✓ Developers more focused on application logic rather than data management
- Easier to model data with flexible schema
 - ✓ Not necessary that every record has same set of attributes
- If possible, treat data as immutable
 - ✓ Keep adding timestamped versions of data values
 - ✓ Avoid human errors by not destroying a good copy

replicated / partitioned storage



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
- Handles high data volume, at very fast rate coming from variety of sources because immutable writes are faster with flexible consistency models
 - ✓ Keep adding data versions with timestamp
 - ✓ Replica updates can keep happening in the background



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

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

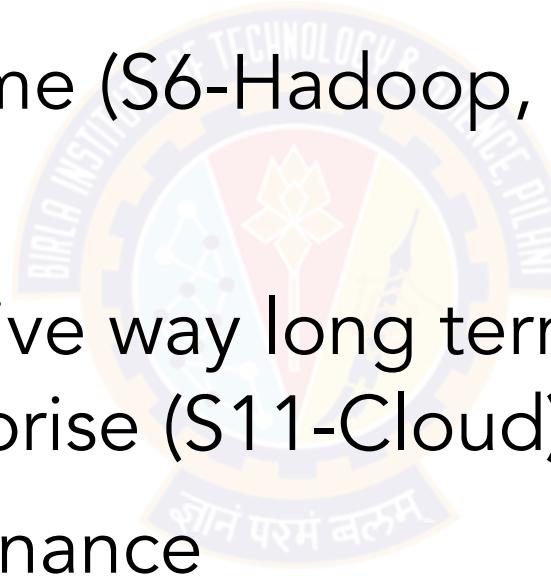
Challenges in Big Data Systems (1)

- Latency issues in algorithms and data storage working with large data sets (S1*-LoR)
- Basic design considerations of Distributed and Parallel systems - reliability, availability, consistency (S2, S3, S4)
- What data to keep and for how long - depends on analysis use case
- Cleaning / Curation of data (S4-Lifecycle)
- Overall orchestration involving large volumes of data (S9)
- Choose the right technologies from many options, including open source, to build the Big Data System for the use cases (S6 onwards)

* Si: Topic is discussed in session i

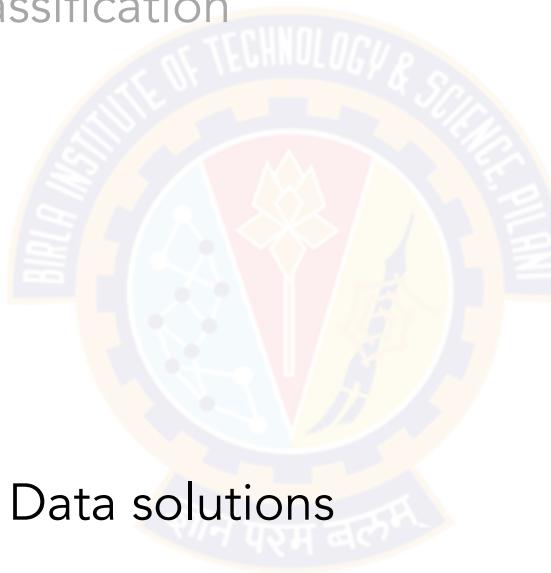
Challenges in Big Data Systems (2)

- Programming models for analysis (S5, S6 - MapReduce, S13-Spark)
- Scale out for high volume (S6-Hadoop, S10-NoSQL, S13-Spark)
- Search (S10-NoSQL)
- Cloud is the cost effective way long term - but need to host Big Data outside the Enterprise (S11-Cloud)
- Data privacy and governance
- Skilled coordinated teams to build/maintain Big Data Systems and analyse data



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
- Why locality of reference and storage organisation matter
- Summary



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

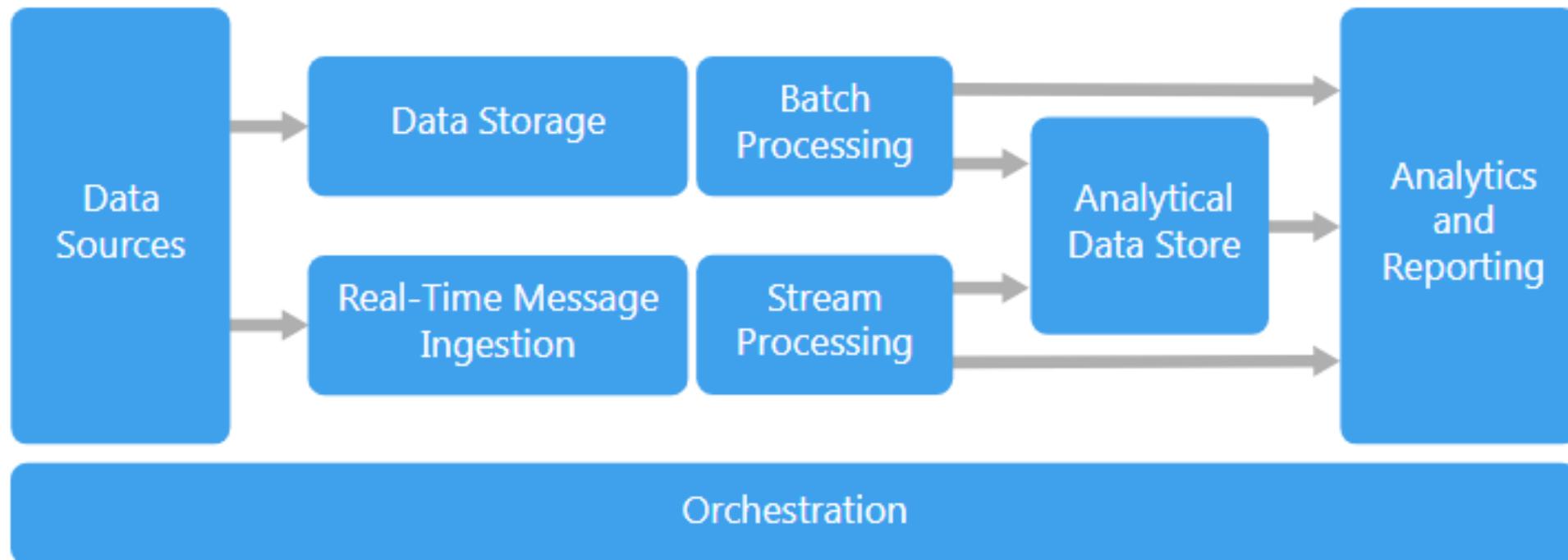
- ✓ Fraud detection from real-time financial transaction data
- ✓ Detect fake news on social media platforms

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](#)

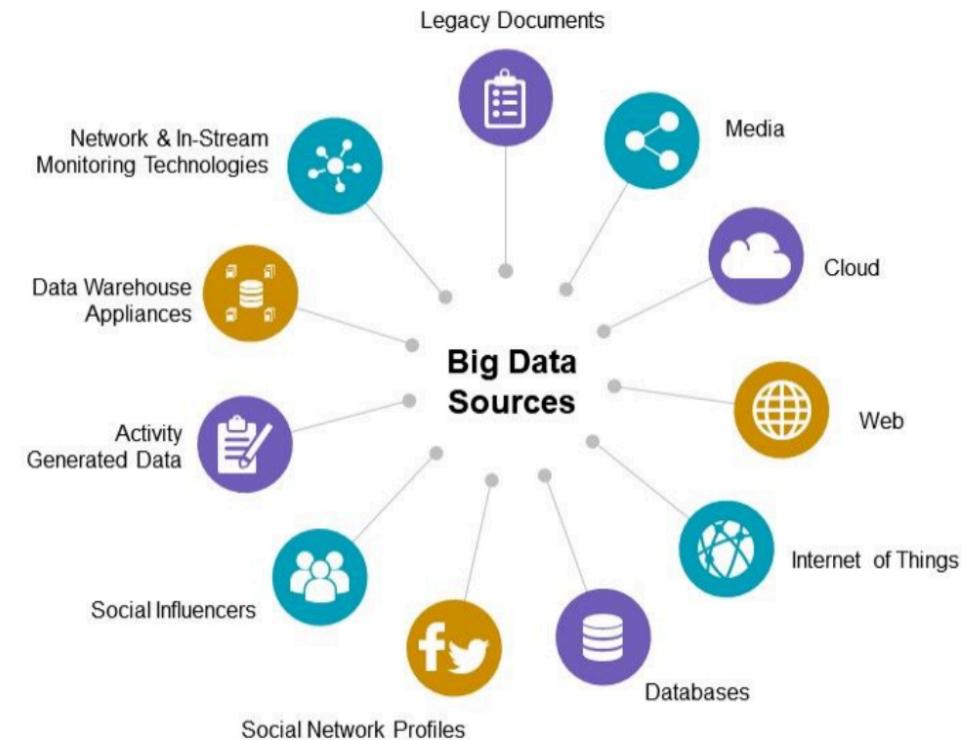
Big Data Systems Components (1)

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.

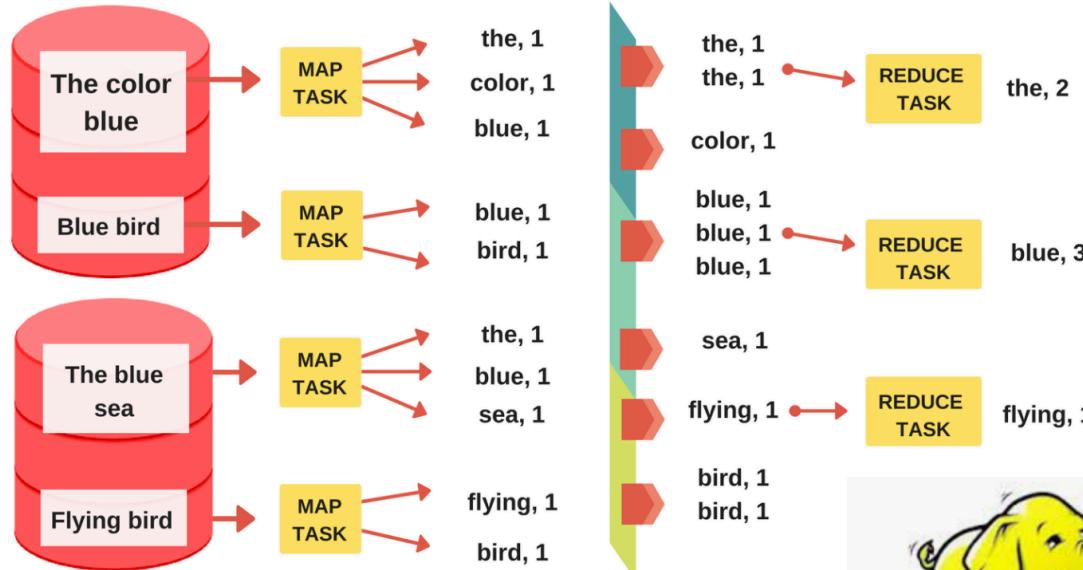


e.g. social data



e.g. medical images

Big Data Systems Components (2)



e.g. search scans on unindexed docs



e.g. credit card transactions for fraud detection

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.

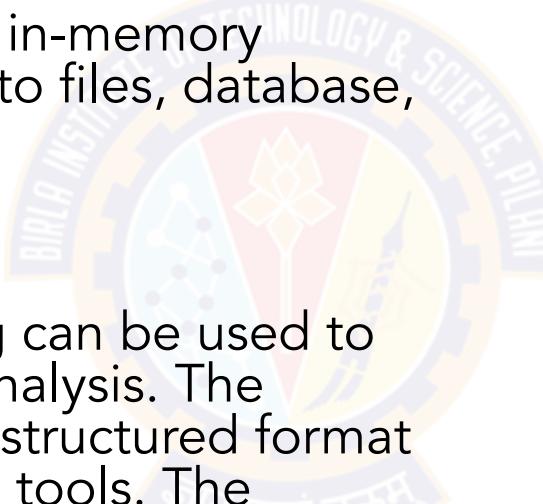
4. Real-time message ingestion

- ✓ 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.



6. Analytical data store

Real-time or batch processing can be used to prepare the data for further analysis. The processed data is 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. fraud detection logic



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

Big Data Systems Components (4)



e.g. weekly management report



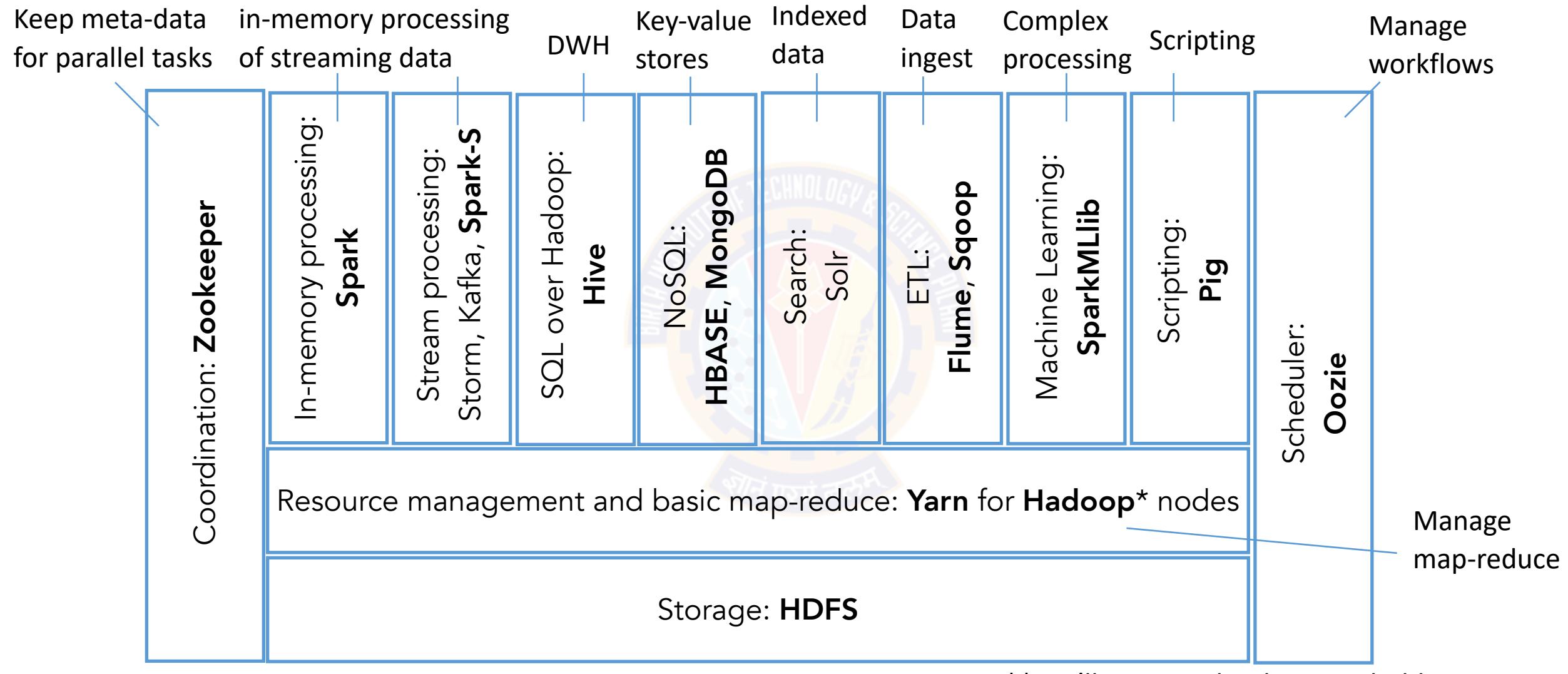
7. Analysis and reporting

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)





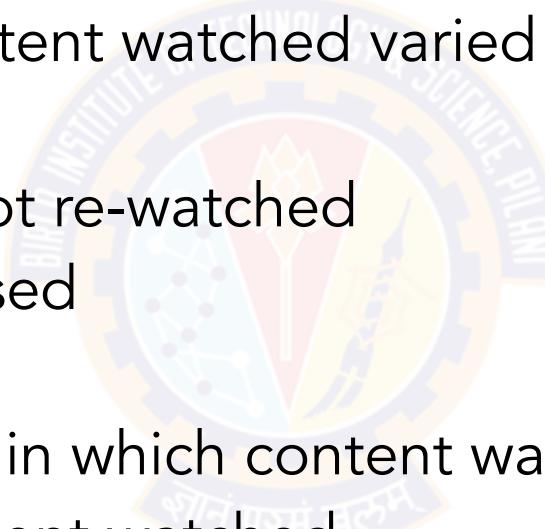
Case Study: Netflix

How Netflix uses Big Data for exponential user growth and retention

(reference: <https://www.clickz.com/how-netflix-uses-big-data-content/228201/>)

Data collected from 150+ million users

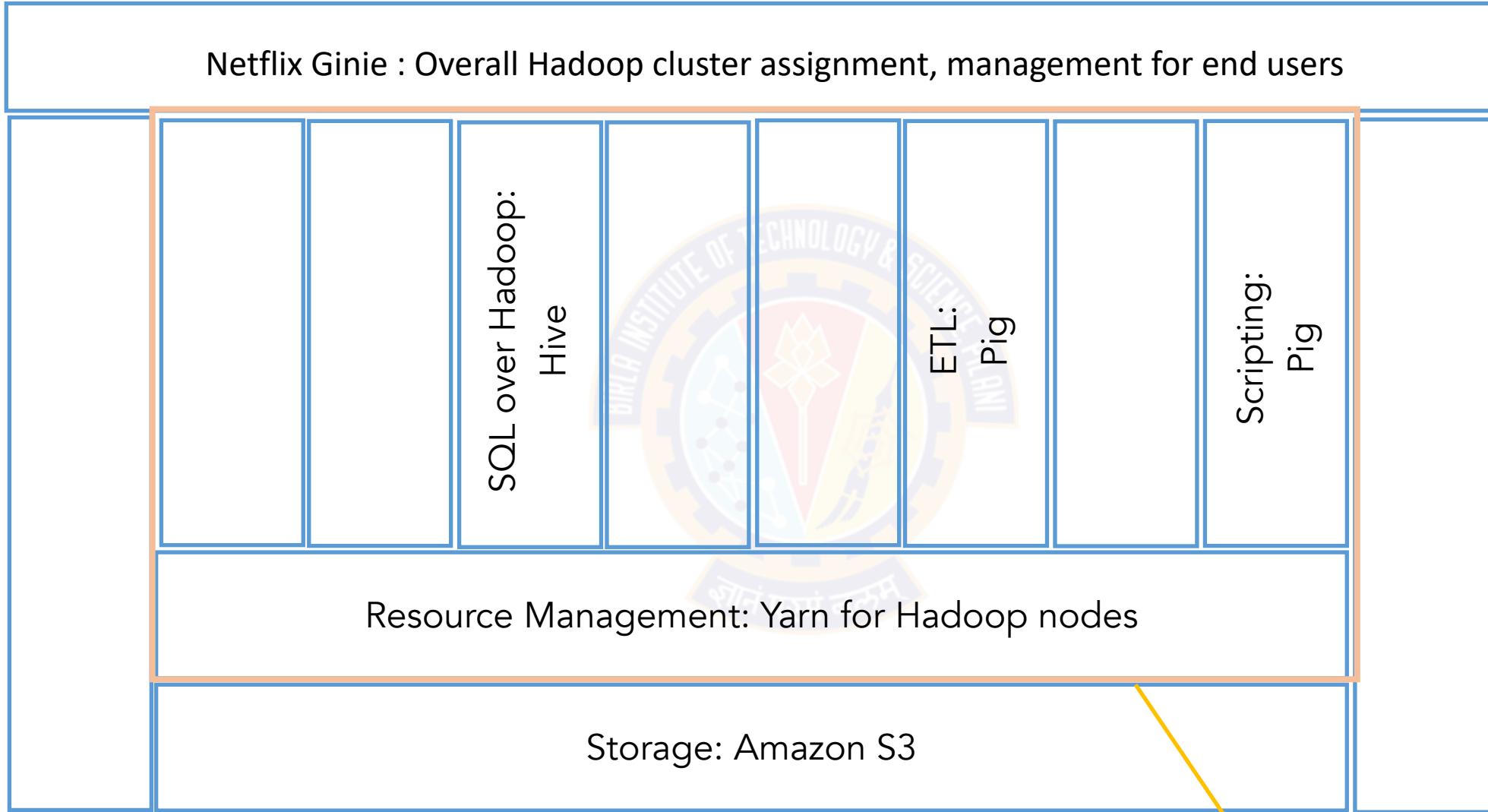
- Date content watched
- The device on which the content was watched
- How the nature of the content watched varied based on the device
- Searches on its platform
- Portions of content that got re-watched
- Whether content was paused
- User location data
- Time of the day and week in which content was watched and how it influences the kind of content watched
- Metadata from third parties like Nielsen
- Social media data from Facebook and Twitter



Recommendation System

- Personalized context ranking
- Content promotion influenced by personal interests and not just what's popular or needs promotion
- Intelligent prioritization of viewed content that users are expected to watch / re-watch - never bore the user
- What type of content is likely to be popular to decide new shows
 - ✓ E.g. "House of Cards" was contracted for 2 seasons without a pilot because analysis said that show will be popular based on analysis of historical viewership data across regions for past content by cast and crew

Big Data System



Amazon EMR (Elastic Map Reduce) - Hadoop on Cloud

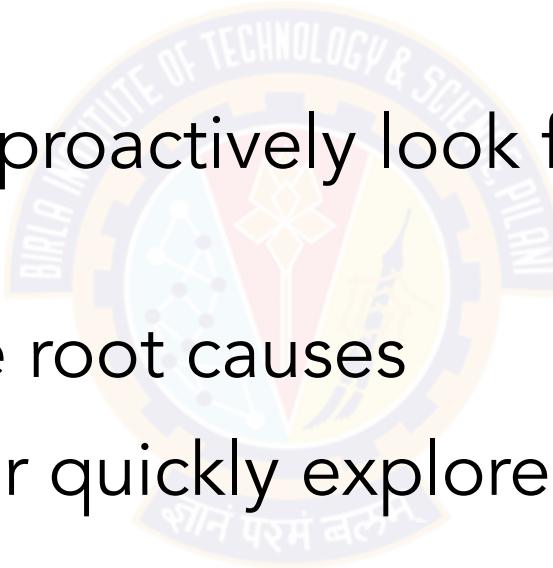


Case Study: IT Ops

Using Big Data tools and architecture for managing IT

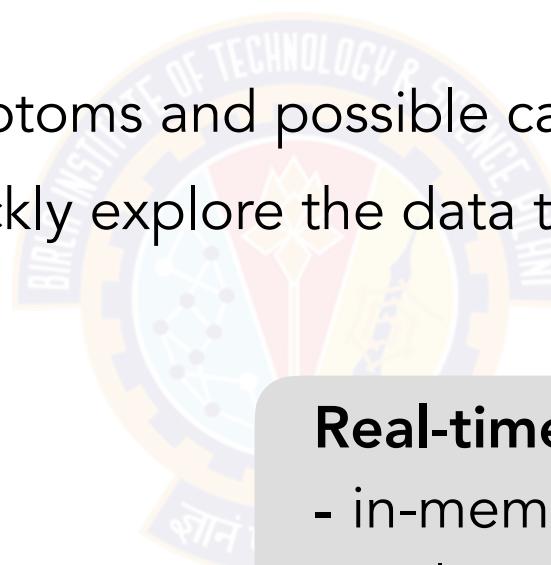
IT Operations Analytics

- 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 localise possible symptoms and possible causes
- Can we help an engineer quickly explore the data to confirm the specific root cause



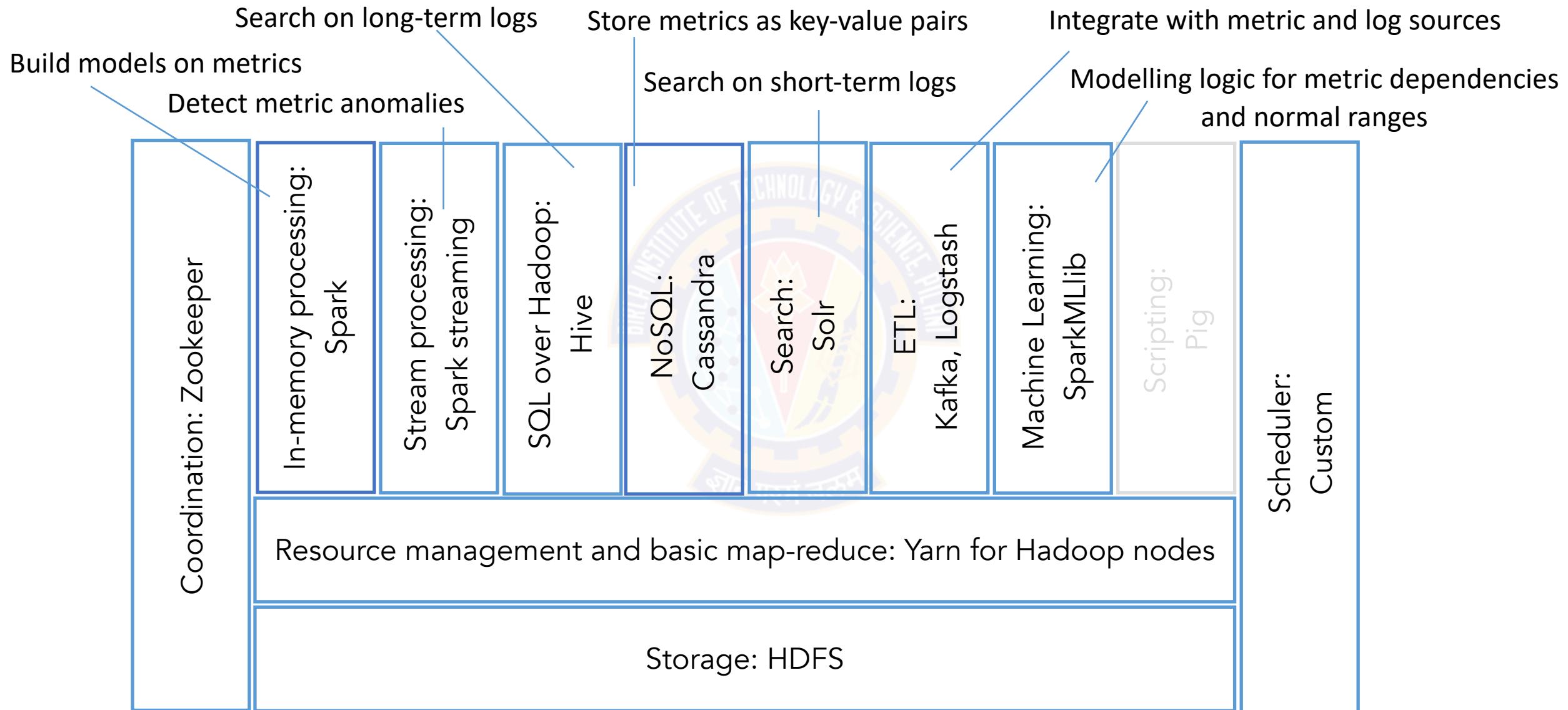
Real-time streaming analysis of metrics

- 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

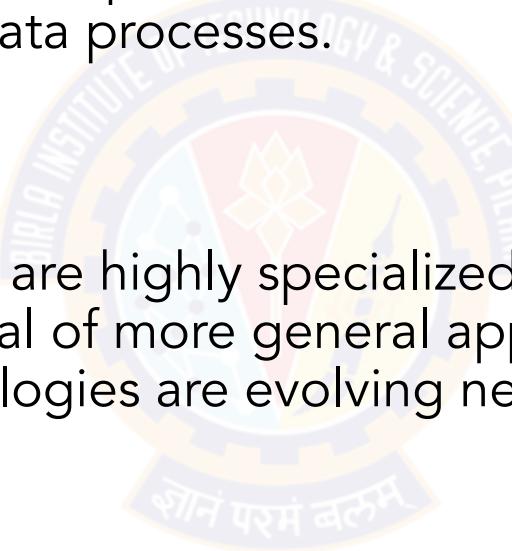
- Consider this architecture style when you need to:
 - ✓ Store and process data in **volumes too large** for a traditional database
 - ✓ Handle **semi-structured** or data with evolving structure - e.g. demographic data with hundreds of attributes
 - ✓ Transform **unstructured data** for analysis and 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

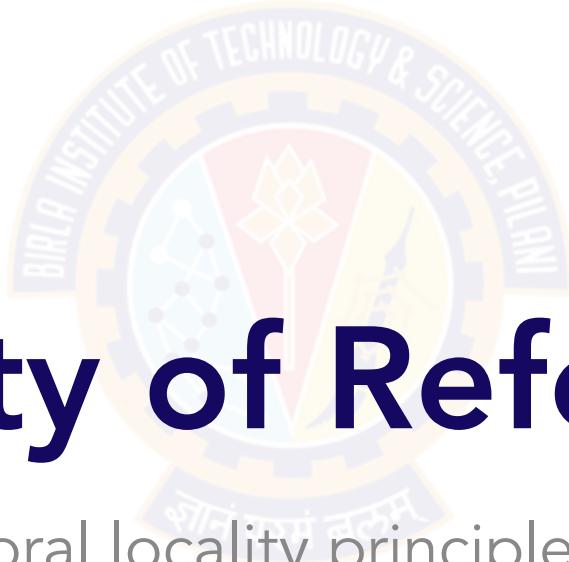
- Complexity
 - ✓ Big data solutions can be extremely complex, with numerous components to handle data ingestion from multiple data sources. It can be challenging to build, test, and troubleshoot big data processes.
- Skillset
 - ✓ Many big data technologies are highly specialized, and use frameworks 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
- Why locality of reference and storage organisation matter



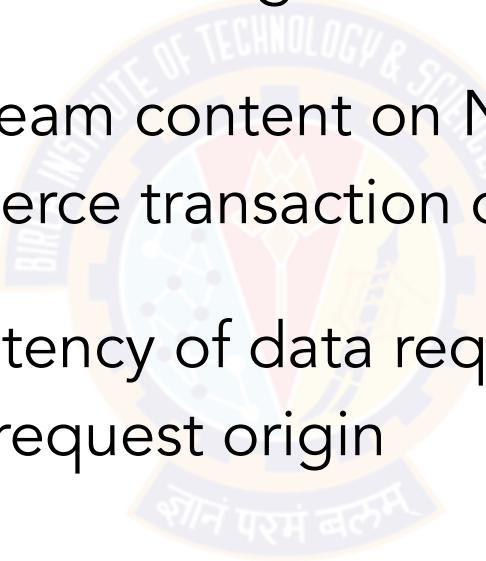


Locality of Reference (LoR)

Spatial and temporal locality principles for processing and storage

Context

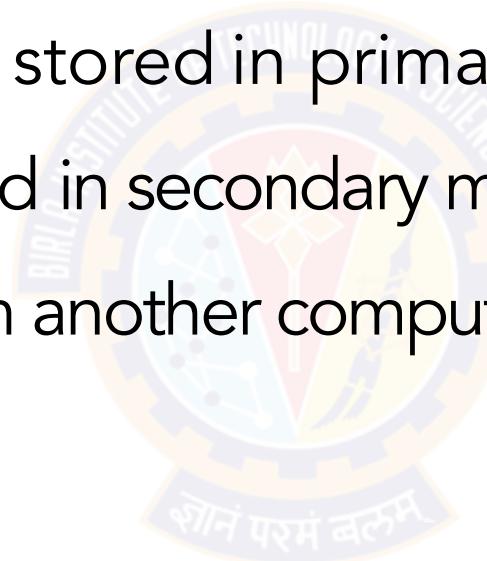
- Big Data Systems need to move large volumes of data
 - ✓ e.g. browse and stream content on Netflix or answer analytical queries on e-commerce transaction data in Amazon
- Is there a way to reduce latency of data requests using locality of reference in the data and request origin



Levels of storage

Data Location – Memory vs Storage vs Network

- Computational Data is stored in primary memory aka memory
- Persistent Data is stored in secondary memory aka Storage
- Remote data access from another computer's memory or storage over network



Cost of access: Memory vs. Storage vs. Network

Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
<u>Main memory reference</u>	<u>100 ns</u>
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
<u>Disk seek</u>	<u>10,000,000 ns</u>
Read 1 MB sequentially from disk	20,000,000 ns
<u>Send packet CA->Netherlands->CA</u>	<u>150,000,000 ns</u>

Reference: <http://www.cs.cornell.edu/projects/ladis2009/talks/dean-keynote-ladis2009.pdf>

Memory hierarchy – motivation

- A memory hierarchy amortizes cost in computer architecture:
 - ✓ fast (and therefore costly) but small-sized memory to
 - ✓ large-sized but slow (and therefore cheap) memory

- **Classic:**

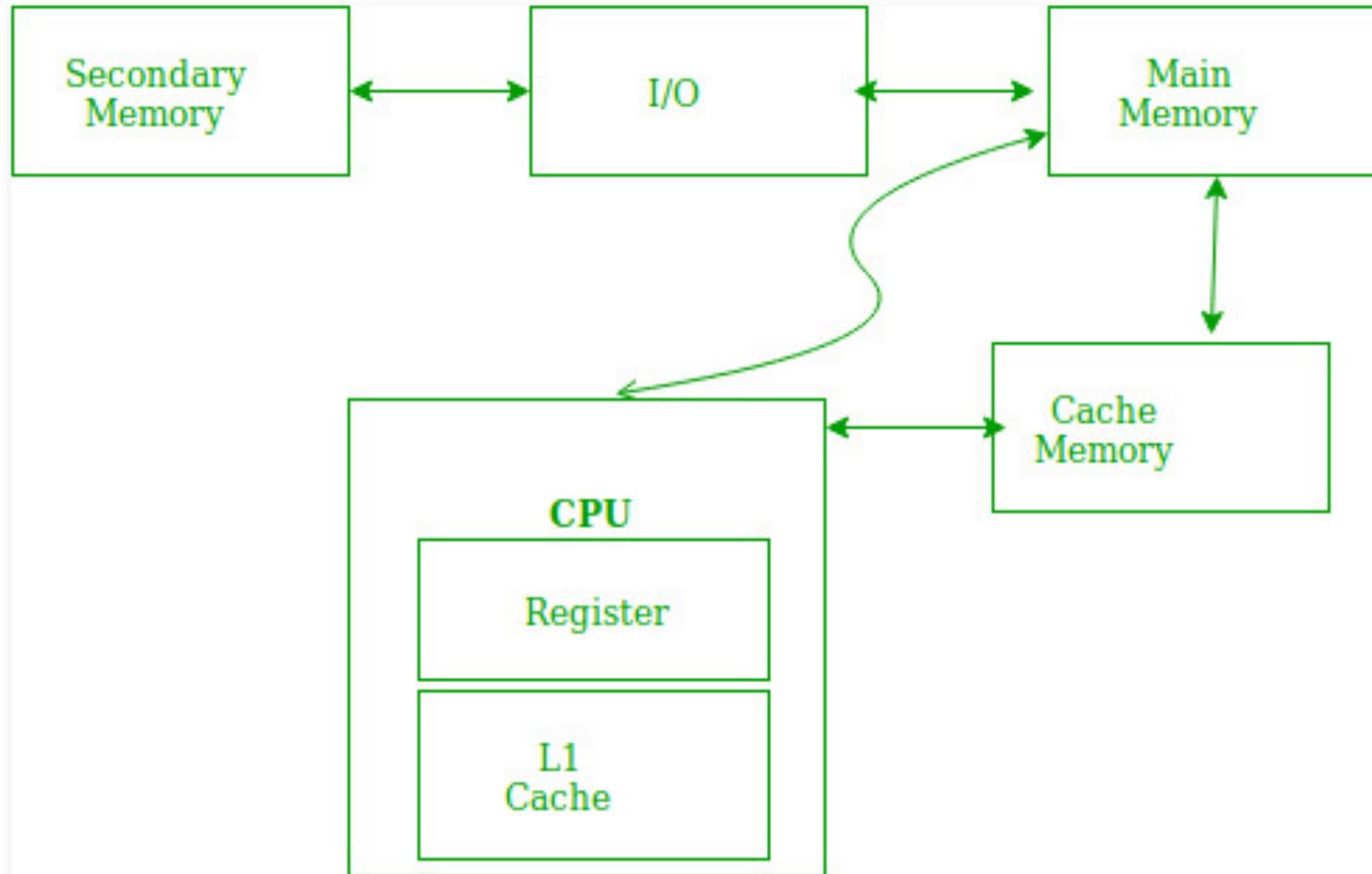


In a BigData search, recent Solr indexed data can be a cache for long term HDFS data.

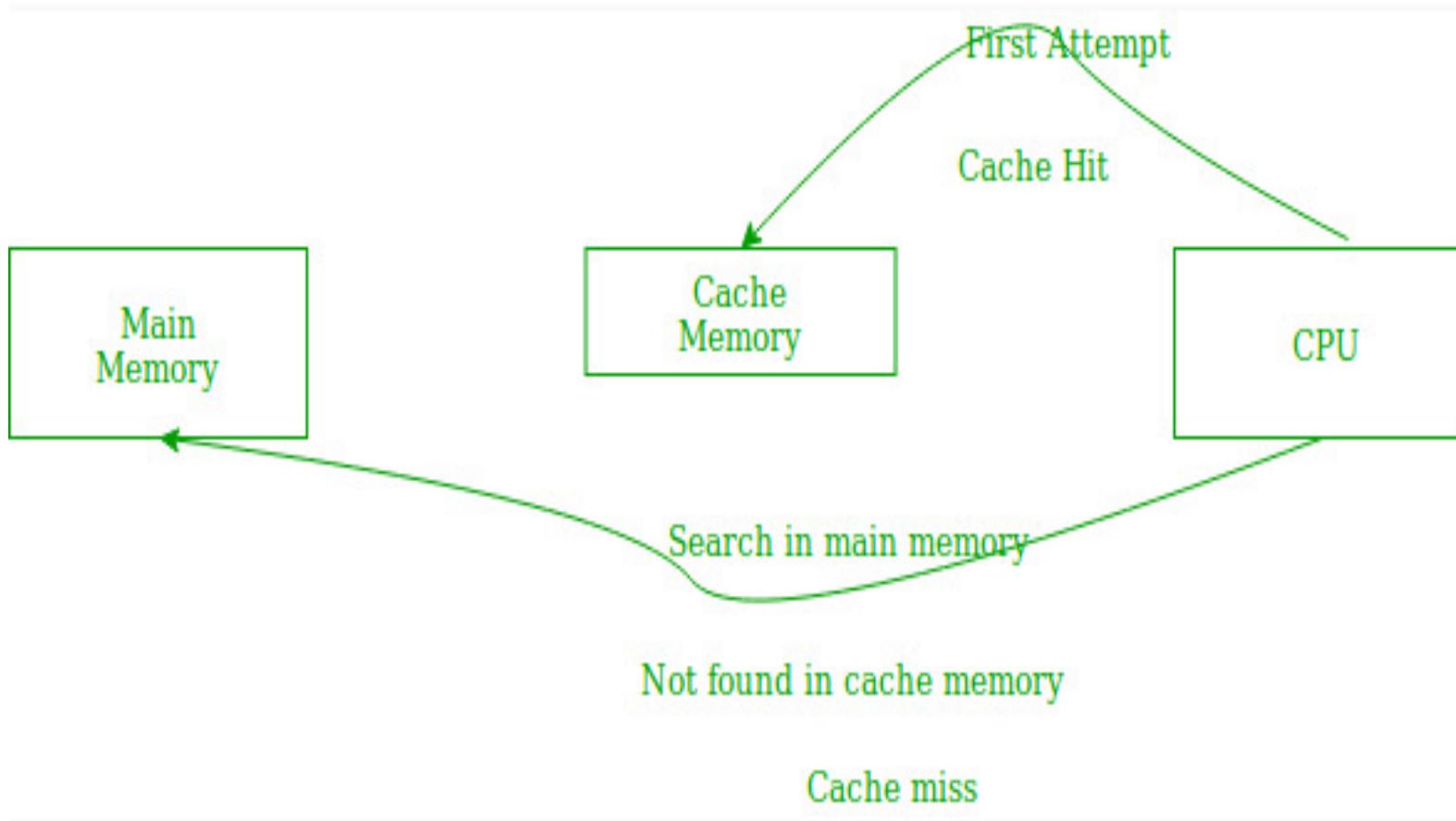
- **Modern:**



Memory hierarchy - access



Memory hierarchy - cache hit/miss



Cache performance

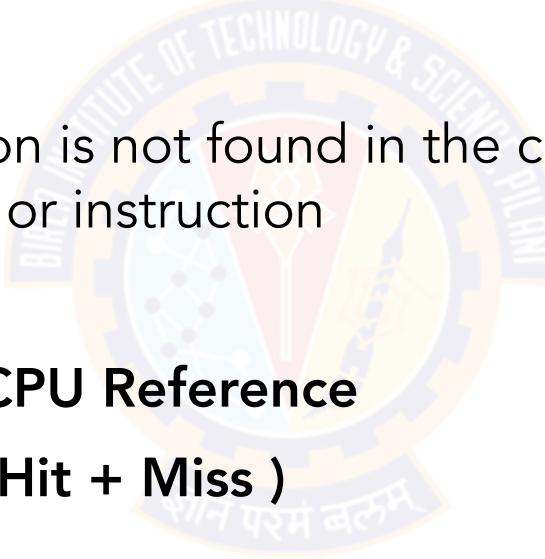
Hit Ratio - The performance of the cache

- Cache hit
 - ✓ When CPU refers to memory and find the data or instruction within the Cache Memory
- Cache miss
 - ✓ If the desired data or instruction is not found in the cache memory and CPU refers to the main memory to find that data or instruction

Hit + Miss = Total CPU Reference

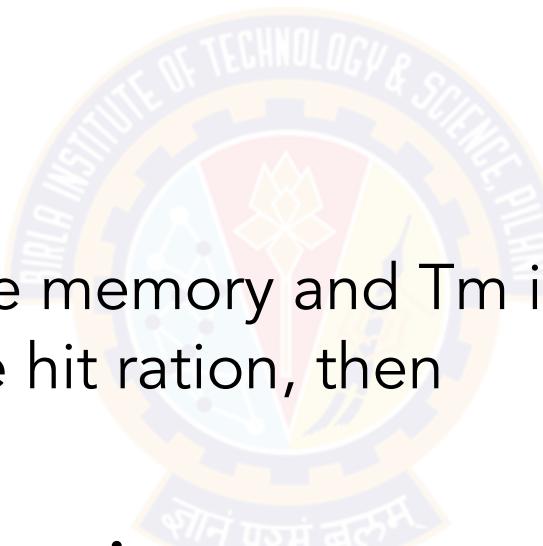
Hit Ratio $h = \text{Hit} / (\text{Hit} + \text{Miss})$

- One can generalise this to any form of cache e.g. movie request from user to nearest Netflix content cache



Access time of memories

- **Average access time** of any memory system consists of two levels:
 - ✓ Cache Memory
 - ✓ Main Memory
- If T_c is time to access cache memory and T_m is the time to access main memory and h is the cache hit ration, then

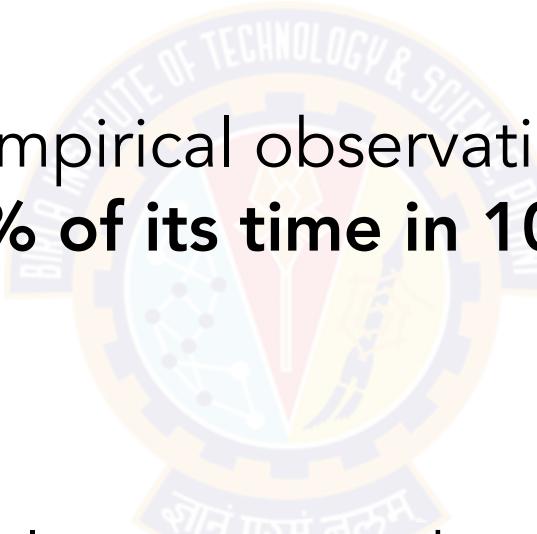


Tavg = Average time to access memory

$$\text{Tavg} = h * T_c + (1-h) * (T_m + T_c)$$

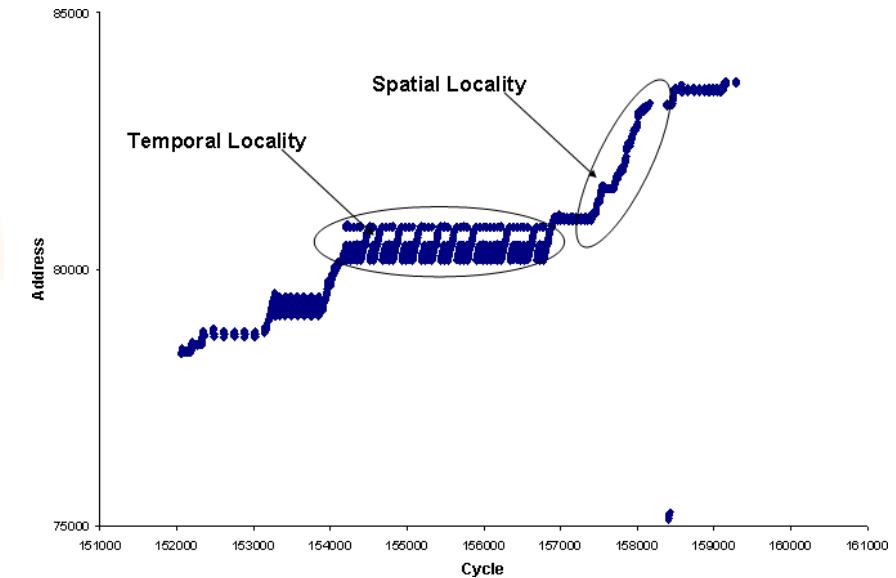
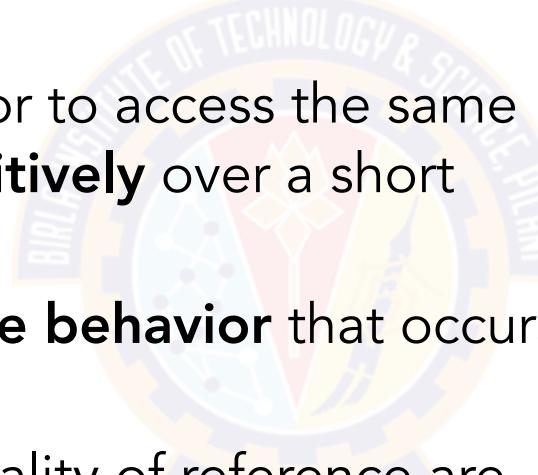
Data reference empirical evidence

- Programs tend to reuse data and instructions they have used recently.
- 90/10 rule comes from empirical observation:
"A program spends 90% of its time in 10% of its code"
- Implication:
 - ✓ Predict with reasonable accuracy what instructions and data a program will use in the near future based on the recent past



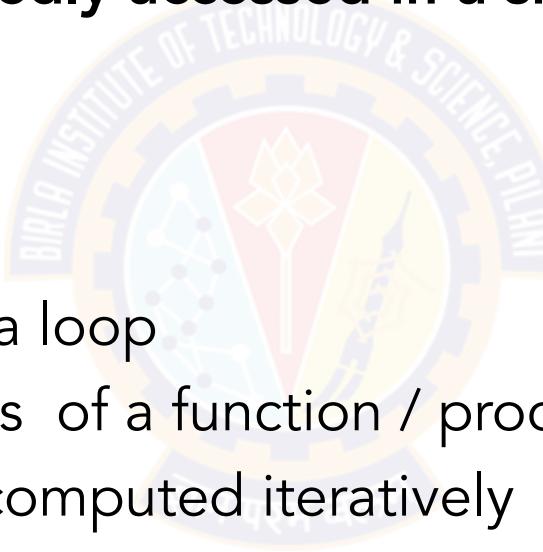
Principle of Locality of Reference (LoR)

1. The **locus of data** access – and hence that of memory references – is small at any point during program execution
2. It is the tendency of a processor to access the same set of memory locations **repetitively** over a short period of time
3. Locality is a type of **predictable behavior** that occurs in computer systems
4. Systems that exhibit strong locality of reference are great candidates for **performance optimization** through the use of techniques such as the caching, prefetching for memory and advanced branch predictors at the pipelining stage of a processor core.



Locality of Reference - Temporal locality

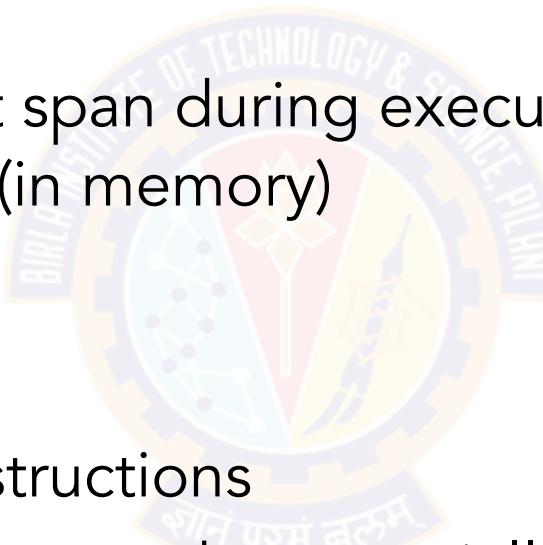
- Data that is accessed (at a point in program execution) is likely to be accessed again in the near future:
i.e. data is likely to be **repeatedly accessed in a short span of time** during execution
- Examples
 1. Instructions in the body of a loop
 2. Parameters / Local variables of a function / procedure
 3. Data (or a variable) that is computed iteratively
 - e.g. a cumulative sum or product
 4. Another user in Netflix in same region will watch the same episode soon
 5. A recent social media post will be viewed soon by other users



```
void swap(int x, int y)
{
    t = x;
    x = y;
    y = t;
}
```

Locality of Reference - Spatial locality

- Data accessed (at a point in program execution) is likely **located adjacent to data** that is to be accessed in near future:
 - ✓ data accessed in a short span during execution is likely to be within a small region (in memory)

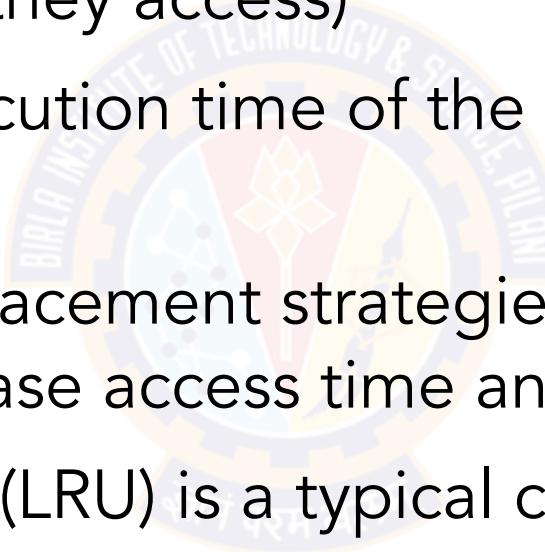


- Examples
 1. A linear sequence of instructions
 2. Elements of an Array (accessed sequentially)
 3. Another user's query will reuse part of the data file brought in for current user's query

```
int sum_array_rows(int marks[8]){
    int i, sum = 0;
    for (i = 0; i < 8; i++)
        sum = sum + marks[i];
    return sum;
}
```

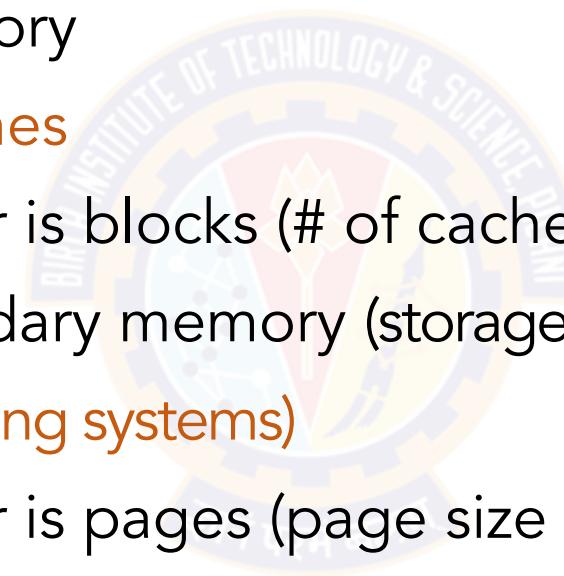
Memory hierarchy and locality of reference (1)

- A **memory hierarchy** is effective only due to **locality** exhibited by programs (and the data they access)
- Longer the range of execution time of the program, larger is the locus of data accesses
- Need to have cache replacement strategies because increasing cache size will also increase access time and increase cost
 - ✓ Least Recently Used (LRU) is a typical cache entry replacement strategy
 - This is an example of temporal locality



Memory hierarchy and locality of reference (2)

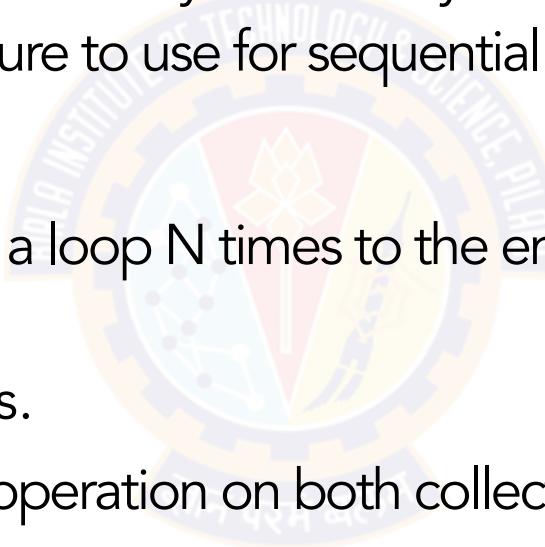
- LOR leads to memory hierarchy at two main interface levels:
 - ✓ Processor - Main memory
 - **Introduction of caches**
 - Unit of data transfer is blocks (# of cache lines)
 - ✓ Main memory - Secondary memory (storage)
 - **Virtual memory (paging systems)**
 - Unit of data transfer is pages (page size is 4 or 8 KB)
- **Fetching larger chunks of data enables spatial locality**



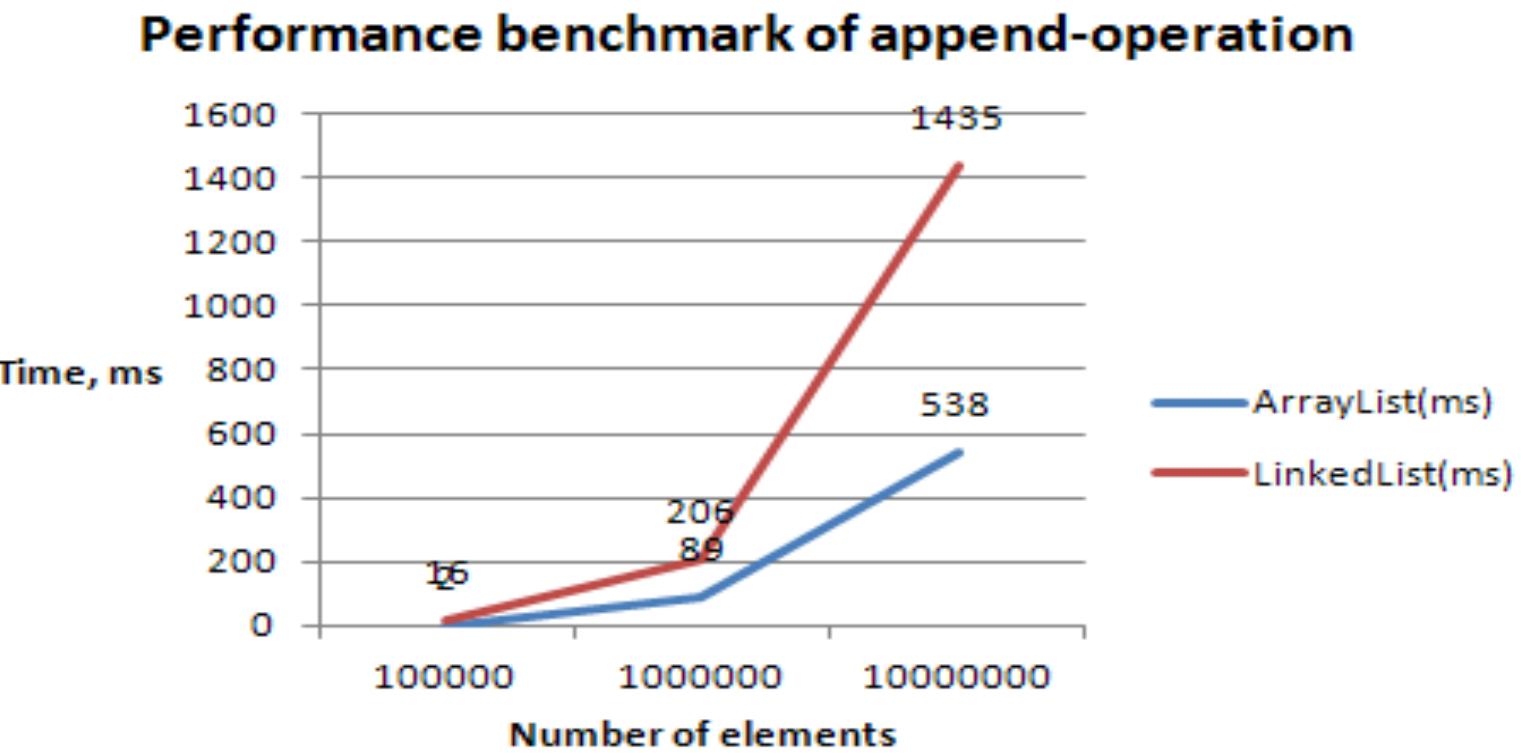
LoR and data structure choices

- In Java, the following 2 classes are available in util package
 - ✓ **LinkedList<Integer>** QuizMarks – a dynamic list of objects
 - ✓ **ArrayList<Integer>** QuizMarks - a dynamic array of objects
 - ✓ Which is a better data structure to use for sequential access performance ?
- Test
 - ✓ Append a single element in a loop N times to the end of a LinkedList. Repeat with ArrayList.
 - ✓ Take average for 100 runs.
 - ✓ The time complexity of the operation on both collections is the same.
- Which one works faster ?

<https://dzone.com/articles/performance-of-array-vs-linked-list-on-modern-comp#:~:text=But%20when%20we%20need%20to,have%20better%20performance%20than%20array.>



LoR and data structure choices (2)



- Analysis - sequential access in **arrays** is faster than on **linked lists** on many machines, because they have a very good spacial locality of reference and thus make good use of data caching.

Locality Example : Matrices addition

- Consider matrices and an operation such as the addition of two matrices:
- Elements $M[i,j]$ may be accessed either row by row or column by column
- Option 1: Add columns within each row (Row Major)

```
for (i=0; i<N; i++)
```

```
    for(j=0; j<N; j++)
```

$$M3[i, j] = M2[i, j] + M1[i, j]$$

- Option 2: Add rows within each column (Column Major)

```
for (j=0; j<N; j++)
```

```
    for(i=0; i<N; i++) •
```

$$M3[i, j] = M2[i, j] + M1[i, j]$$

Both options have same time complexity.
Why is one faster ?

Locality Analysis (Row major)

M [I] [J]	J=0	J=1	J=2	J=3
I=0	W[0] miss	W[1] hit	W[2] hit	W[3] hit
I=1	W[4] Miss	W[5] Hit	W[6] Hit	W[7] hit
I=2	W[8] miss	W[9] hit	W[10] hit	W[11] hit
I=3	W[12] miss	W[13] hit	W[14] hit	W[15] hit

Locality Analysis (Column major)

M[I] [J]	J=0	J=1	J=2	J=3
I=0	W[0] miss	W[1] miss	W[2] miss	W[3] miss
I=1	W[4] Miss	W[5] miss	W[6] miss	W[7] miss
I=2	W[8] miss	W[9] miss	W[10] miss	W[11] miss
I=3	W[12] miss	W[13] miss	W[14] miss	W[15] miss

Big Data: Storage organisation matters

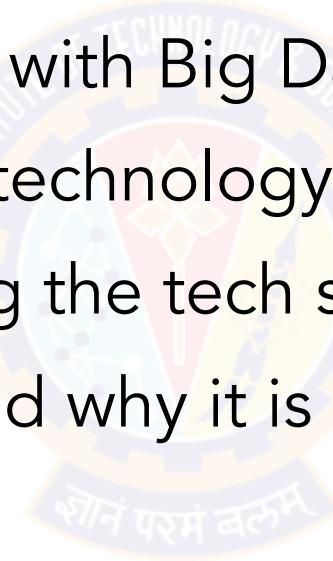
Example

- We need to build a prediction model of sales for various regions
- There are many attributes for a region and “total sales” is the metric used
- Suppose the database is “columnar” (Column major data organisation)
 - ✓ Will exhibit high spatial locality and hit rate because data blocks will fetch blocks of total sales column and not other unnecessary columns
 - ✓ Will improve speed of modelling logic
- Columnar storage is common in most Big Data Systems to run analysis and queries that focus on specific attributes at a time for searching, aggregating, modelling etc.



Summary

- Why modern Enterprises and new age applications are data-centric
- Challenges with existing data systems
- Advantages and challenges with Big Data systems
- High level architecture and technology ecosystem
- Some real applications using the tech stack
- Data locality of reference and why it is useful for Big Data Systems





Next Session:

Parallel and Distributed Processing