Unit -2 Big Data Technology Foundations

Big Data (CS-3032)

Kalinga Institute of Industrial Technology
Deemed to be University
Bhubaneswar-751024

School of Computer Engineering

Strictly for internal circulation (within KIIT) and reference only. Not for outside circulation without permission

3 Credit

Lecture Note

Course Contents

Sr #	Major and Detailed Coverage Area	Hrs
2	**Big Data Technology Foundations**	8
	Exploring the Big Data Stack, Data Sources Layer, Ingestion Layer, Storage Layer, Physical Infrastructure Layer, Platform Management Layer, Security Layer, Monitoring Layer, Analytics Engine, Visualization Layer, Big Data Applications, Virtualization. Introduction to Streams Concepts – Stream data model and architecture – Stream Computing, Sampling data in a stream – Filtering streams, Counting distinct elements in a stream.	

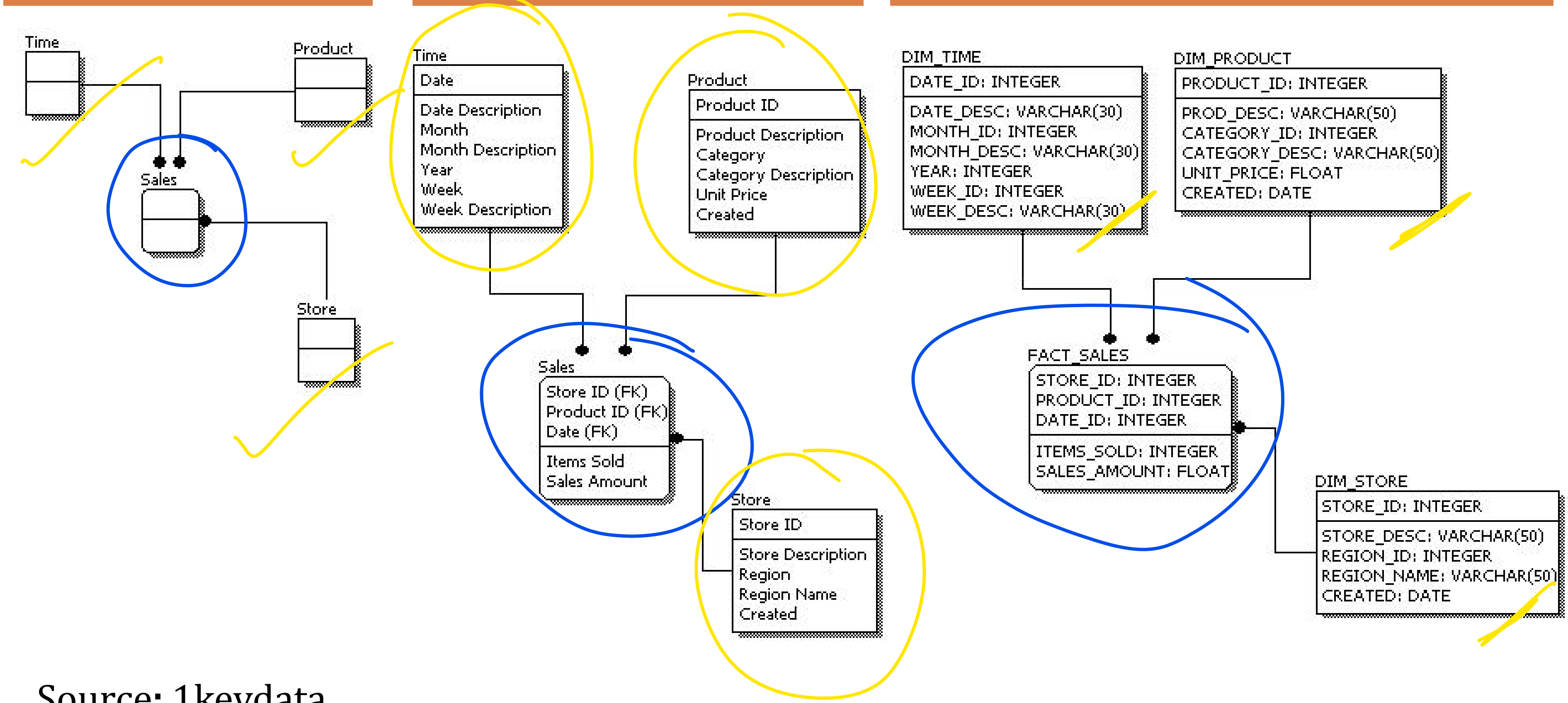
Data Architecture

In information technology, data architecture is composed of models, policies, rules or standards that govern which data is collected, and how it is stored, arranged, integrated, and put to use in data systems and in organizations. A few basic concepts in data architecture:

- ❑ Conceptual data model—shows data entities such as customer, product and transaction, and their semantics.
- ❑ Logical model—defines the data in as much detail as possible, including relations between data elements, but without considering how data is stored or managed.
- ❑ Physical data model—defines how the data is represented and stored, for example in a flat file, database, data warehouse, key-value store

Data Model

Conceptual Data Model | *Logical Data Model* | *Physical Data Model*



Time

Product

Sales

Store

Time
Date
Date Description
Month
Month Description
Year
Week
Week Description

Product
Product ID
Product Description
Category
Category Description
Unit Price
Created

Sales
Store ID (FK)
Product ID (FK)
Date (FK)
Items Sold
Sales Amount

Store
Store ID
Store Description
Region
Region Name
Created

DIM_TIME
DATE_ID: INTEGER
DATE_DESC: VARCHAR(30)
MONTH_ID: INTEGER
MONTH_DESC: VARCHAR(30)
YEAR: INTEGER
WEEK_ID: INTEGER
WEEK_DESC: VARCHAR(30)

DIM_PRODUCT
PRODUCT_ID: INTEGER
PROD_DESC: VARCHAR(50)
CATEGORY_ID: INTEGER
CATEGORY_DESC: VARCHAR(50)
UNIT_PRICE: FLOAT
CREATED: DATE

FACT_SALES
STORE_ID: INTEGER
PRODUCT_ID: INTEGER
DATE_ID: INTEGER
ITEMS_SOLD: INTEGER
SALES_AMOUNT: FLOAT

DIM_STORE
STORE_ID: INTEGER
STORE_DESC: VARCHAR(50)
REGION_ID: INTEGER
REGION_NAME: VARCHAR(50)
CREATED: DATE

Source: 1keydata

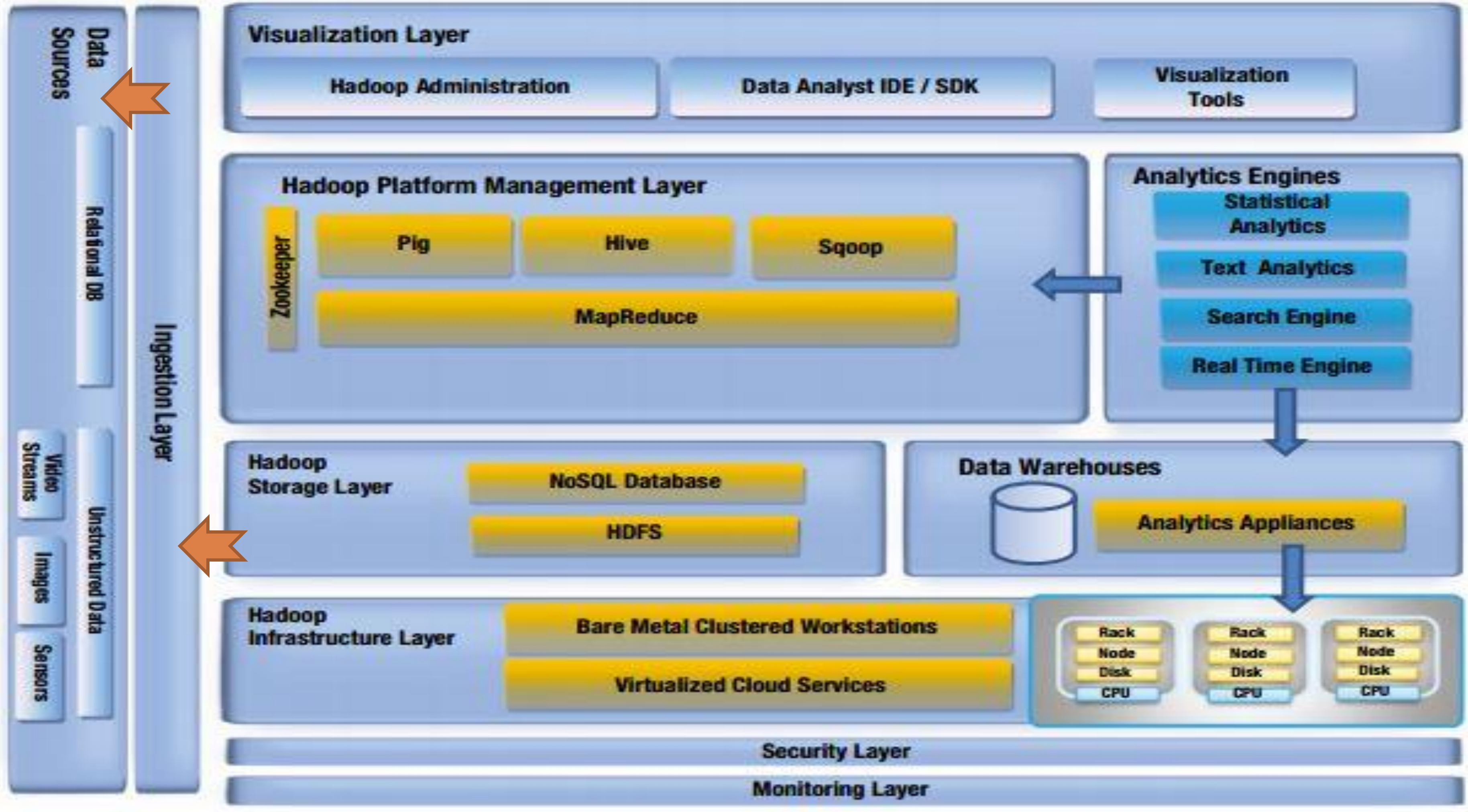
Exploring the Big Data Stack

The first step in the process of designing any data architecture is to create a model that should give a complete view of all required elements. Although, initially, creating a model may seem to be a time-consuming task, however, it can save a significant amount of time, effort, and rework during the subsequent implementations.

Big Data analysis also needs the creation of a model or architecture, commonly known as the Big Data architecture. Such architecture of the big data environment must fulfill all the foundational requirements and must be able to perform the following key functions:

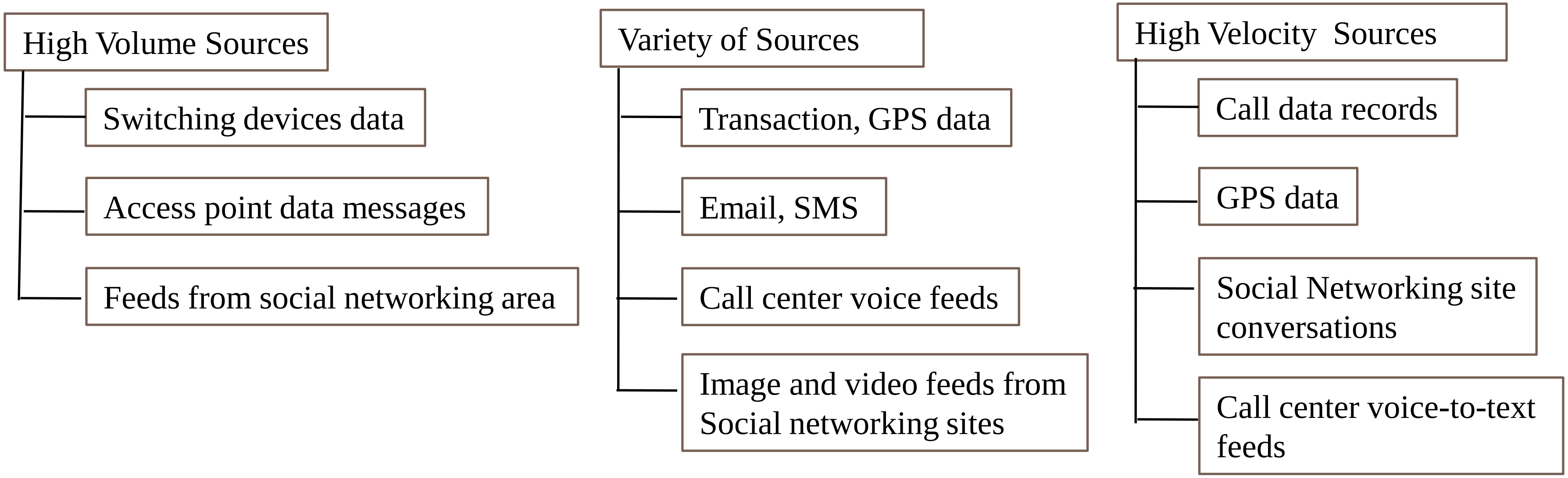
☑ Capturing data from different data sources
☑ Cleaning and integrating data of different types of format
☑ Storing and organizing data
☑ Analyzing data
☑ Identifying relationship and patterns
☑ Deriving conclusions based on the data analysis results

Big Data Architecture Layers



Data Sources Layer

The basic function of the data sources layer is to absorb and integrate the data from various sources, at varying velocity and in different formats. It includes everything from sales records, customer database, feedback, social media channels, marketing list, email archives and any data gleaned from monitoring or measuring aspects of operations. Before this data is considered for big data stack, differentiation is required between the noise and relevant information. Example of different data sources the telecom industry obtains its data:

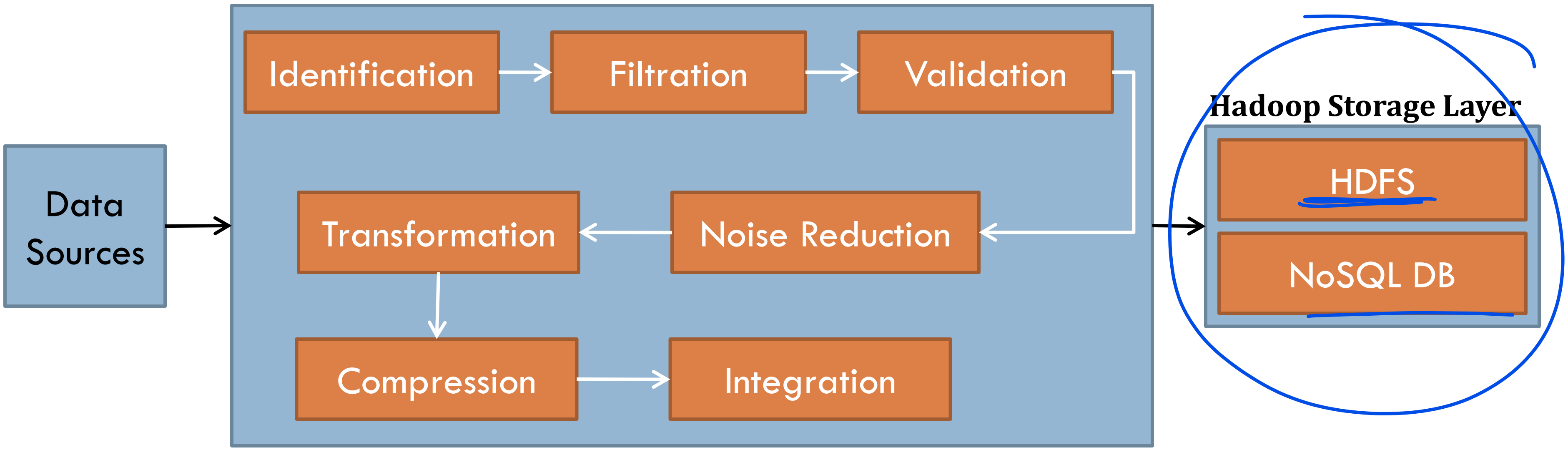


- High Volume Sources
  - Switching devices data
  - Access point data messages
  - Feeds from social networking area
- Variety of Sources
  - Transaction, GPS data
  - Email, SMS
  - Call center voice feeds
  - Image and video feeds from Social networking sites
- High Velocity Sources
  - Call data records
  - GPS data
  - Social Networking site conversations
  - Call center voice-to-text feeds

Ingestion Layer

2M

It absorb the huge inflow of data and separates noise from relevant information. It handle huge volume, high velocity, and a variety of data. It validates, cleanses, transforms, reduces, and integrates the unstructured data into the Big Data stack for further processing. In this layer, the data passes through the following stages.



I F V N T C I

Ingestion Layer cont'd

In the ingestion layer, the data passes through the following stages:

1. **Identification** – At this stage, data is categorized into various known data formats or even unstructured data is assigned with default formats.
2. **Filtration** – At this stage, the information relevant for the enterprise is filtered on the basis of the Enterprise **Master Data Management** (MDM) repository. MDM is a comprehensive method of enabling an enterprise to link all of its critical non-transactional data to a common point of reference. Example – One probably have an address book on both phone and email system and would like to keep the contact details the same on both systems. You also probably use Facebook and would like to add Facebook friends as contacts in your address book. Usually this is viewed in terms of synchronization between the different systems and various software methods are available to enable this. MDM solution for contact management is a single version of the address book is maintained as the master. Different views of the master address book are able to be synchronized to any of the linked address book applications. Every time a contact is added to any of the linked address books it is matched against the master to check whether the contact already exists.

Ingestion Layer cont'd

3. **Validation** – At this stage, the filtered data is analyzed against MDM metadata.
4. **Noise Reduction**– At this stage, data is cleansed by removing the noise and minimizing the related disturbances.
5. **Transformation** - At this stage, data is split or combined on the basis of its type, contents, and the requirements of the organizations.
6. **Compression** - At this stage, the size of the data is reduced without affecting its relevance for the required process and it guarantees of no adverse affect to the analysis result.
7. **Integration** - At this stage, the refined dataset is integrated with the Hadoop storage layer, which primarily consists of HDFS (Hadoop Distributed File System) and NoSQL databases.

Storage Layer

It is the place where Big Data lives. Hadoop is an open source framework used to store large volumes of data in a distributed manner across multiple machines. The Hadoop storage layer supports fault-tolerance and parallelization, which enable high-speed distributed processing algorithms to execute over large-scale data. Two major component of Hadoop: a scalable **Hadoop Distributed File System** that can support petabytes of data and a **MapReduce** Engine that can computes results in batches. HDFS stores data in the form of block of files and follows the write-once-read-many model to access data from these blocks of files. However, apart from files, different types of database are also used. However, storage requirement can be addressed by a single concept known as Not Only SQL (NoSQL) databases. Hadoop has its own database, known as Hbase, but others including Amazon's DynamoDB, MongoDB, AllegroGraph, Cassandra (used by Facebook) and IndefiniteGraph are popular too.

SQL vs. NoSQL

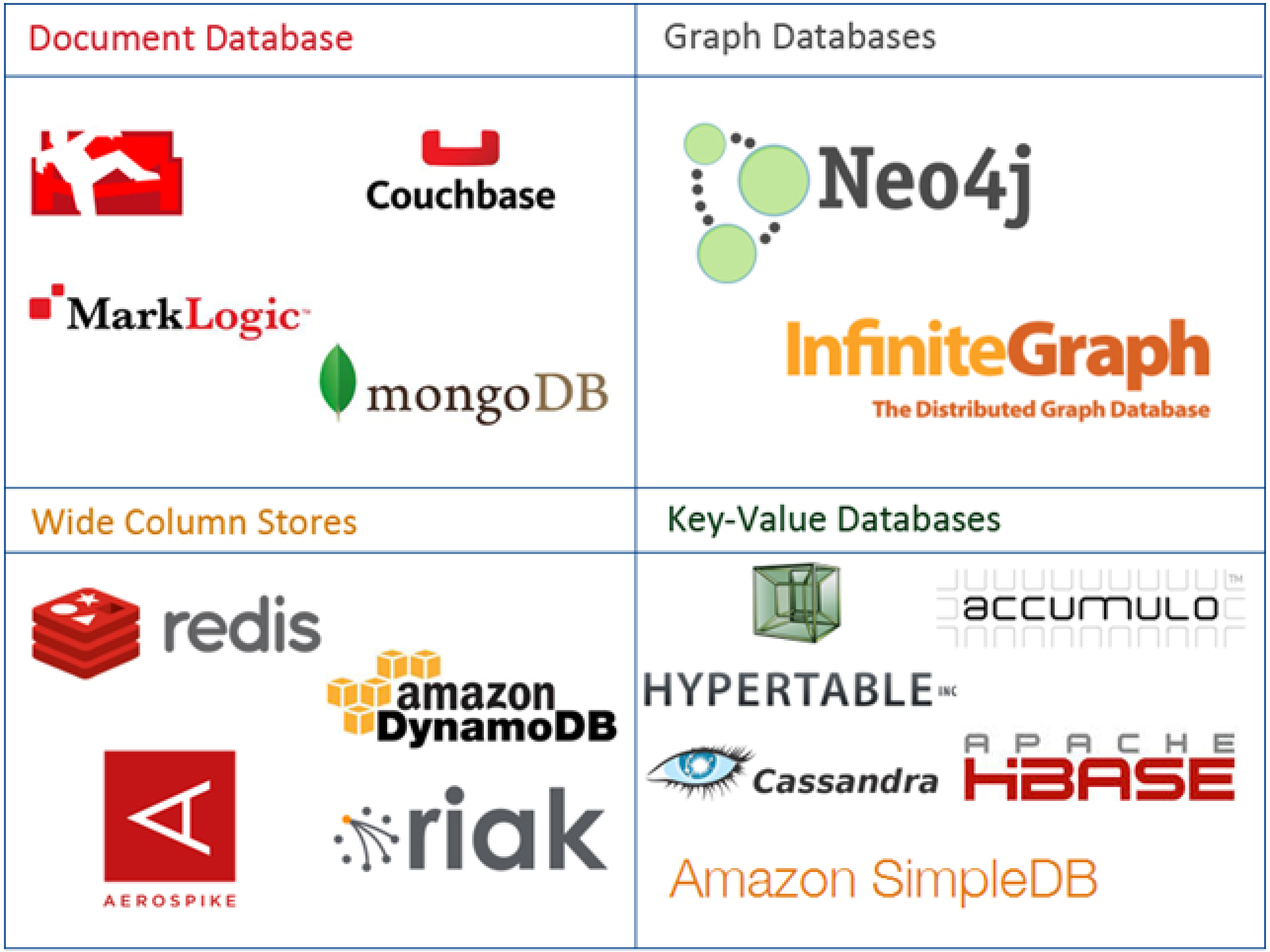
4M

SQL	NoSQL
1 SQL databases are primarily called as Relational Databases (RDBMS)	1 NoSQL database are primarily called as non-relational or distributed database.
2 SQL databases are table based databases i.e. represent data in form of tables which consists of n number of rows of data	2 NoSQL databases are document based, key-value pairs, graph databases or wide-column stores. It do not have standard schema definitions which it needs to adhered to.
3 Predefined schema for structured data	3 Dynamic schema for unstructured data
Vertically scalable wherein databases are scaled by increasing the horse-power of the hardware.	Horizontally scalable wherein databases are scaled by increasing the databases servers in the pool of resources to reduce the load.
Emphasizes on ACID properties (Atomicity, Consistency, Isolation and Durability)	Follows Brewers CAP theorem (Consistency, Availability and Partition tolerance)

SQL vs. NoSQL cont'd

SQL	NoSQL
Not best fit for hierarchical data storage	Fits better for the hierarchical data storage as it follows the key-value pair way of storing data similar to JSON data. highly preferred for large data set
Good fit for the complex query intensive environment	Not good fit for complex queries. On a high-level, NoSQL don't have standard interfaces to perform complex queries, and the queries themselves in NoSQL are not as powerful as SQL query language.
Examples - MySql, Oracle, Sqlite, Postgres and MS-SQL	Examples - MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb

Different NoSQL Databases



Document Database	Graph Databases
Couchbase MarkLogic mongoDB	Neo4j InfiniteGraph The Distributed Graph Database
Wide Column Stores	**Key-Value Databases**
redis amazon DynamoDB AEROSPIKE riak	accumulo HYPERTABLE INC Cassandra APACHE HBASE Amazon SimpleDB

Source: http://www.aryannava.com

Physical Infrastructure Layer

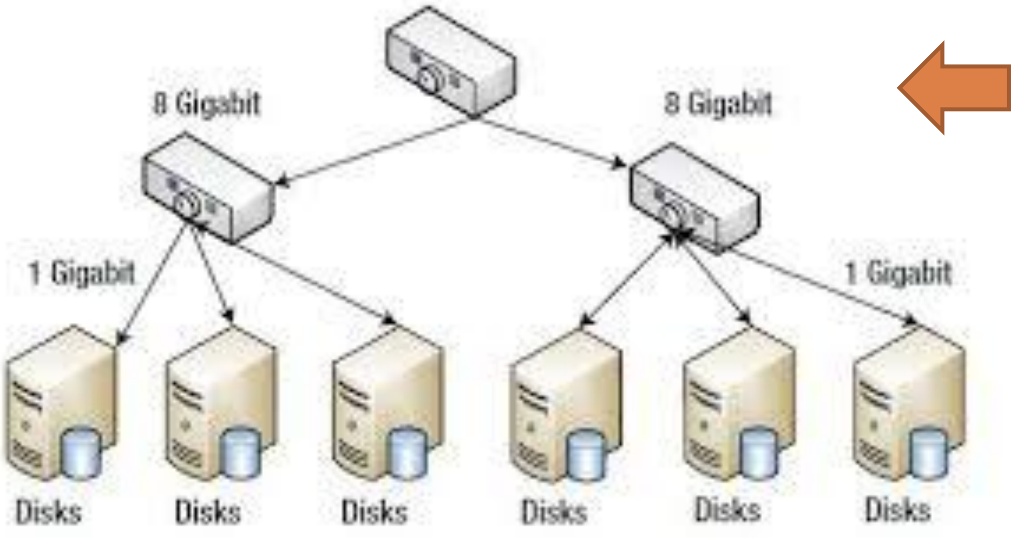
PASFC

Principles on which Big Data implementation is based:

1. **Performance** – High-end infrastructure is required to deliver high performance with low latency (the total time taken by a packet to travel from one node to another node). It is measured end to end, on the basis of a single transaction or query request. It would be rated high if the total time taken in traversing a query request is low.
2. **Availability** – The infrastructure setup must be available at all times to ensure nearly a 100 percent uptime guarantee of service.
3. **Scalability** - The infrastructure must be scalable enough to accommodate varying storage and computing requirements. It must be capable enough to deal with any unexpected challenges.
4. **Flexibility** – Flexible infrastructures facilitate adding more resources to the setup and promote failure recovery.
5. **Cost** – Affordable infrastructure must be adopted including hardware, networking and storage requirements. Such parameters must be considered from the overall budget and trade-offs can be made, where necessary.

Physical Infrastructure Layer cont'd

So it can be concluded that a robust and inexpensive physical infrastructures can be implemented using Big Data. This requirement is handled by the Hadoop physical infrastructure layer. This layer is based on distributed computing model, which allows the physical storage of data in many different locations by linking item through networks and the distributed file systems. It also has to support redundancy of data. This layer takes care of the hardware and network requirements and can provide a virtualized cloud environment or a distributes grid of commodity servers over a fast gigabit network. Hardware topology used for Big Data Implementation is as follows.



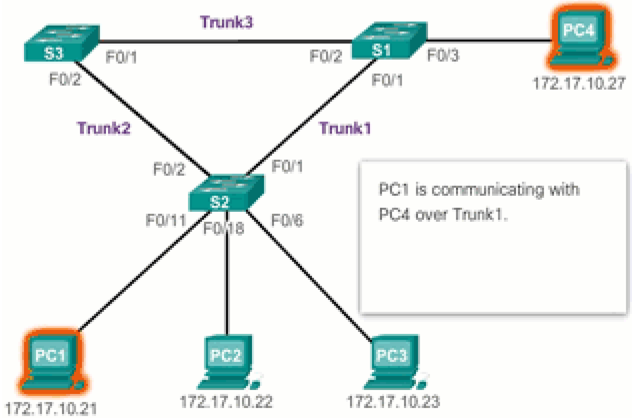
The main components of a Hadoop infrastructure:

1. n commodity servers (8-corem 24GBs RAM, 4 to 12 TBs)

2. 2-level network (20 to 40 nodes per rack)

Physical Infrastructure Design Consideration

1. **Physical Redundant Networks** – In the Big data environment, networks should be redundant and capable of accommodating the anticipated volume and velocity of the inbound and outbound data in case of heavy network traffic. The strategy must be prepared for improving their network performance to handle the increase in the volume, velocity, and variety of data.



PC1 is communicating with PC4 over Trunk1.

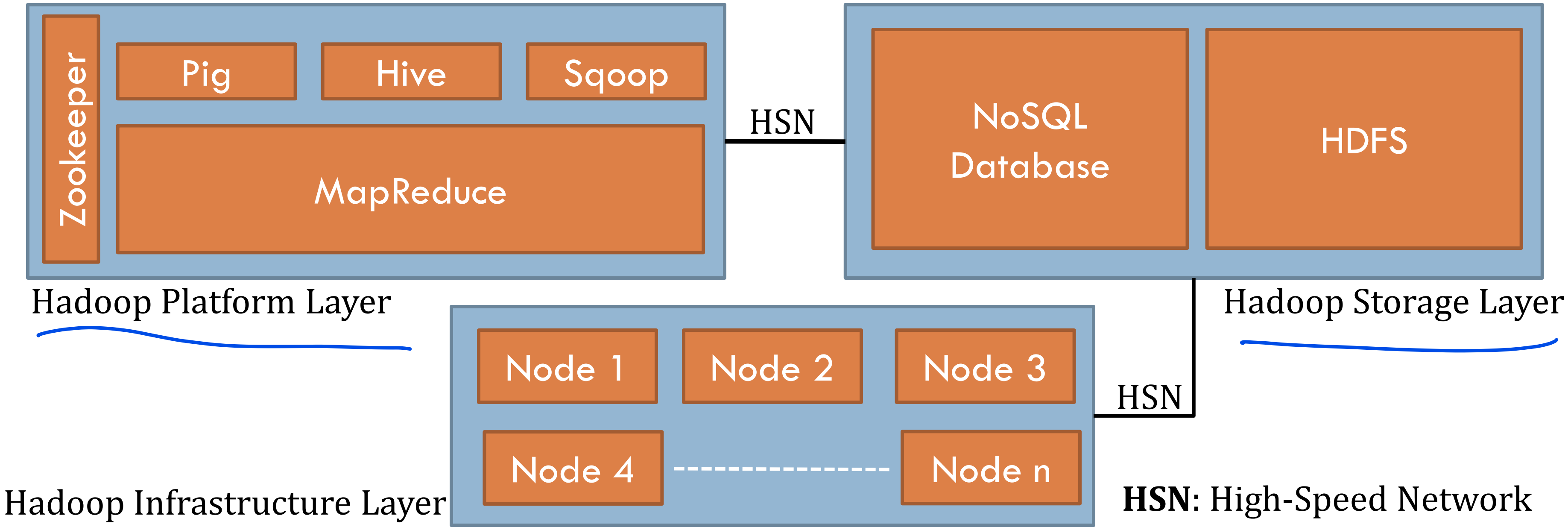
Network redundancy is a process through which additional or alternate instances of network devices, equipment and communication mediums are installed within network infrastructure. It is a method for ensuring network availability in case of a network device or path failure and unavailability.

Physical Infrastructure Design Consideration cont’d

2. **Managing Hardware: Storage and Servers** – Hardware resources for storage and servers must have sufficient speed and capacity to handle all expected types of Big Data. If slow servers are connected to high-speed networks, the slow performance of the servers will be little use and can at a times also become a bottleneck.
3. **Infrastructure Operations** - Proper management of data handling operations provides a well-managed environment, which in turn gives the greatest levels of performance and flexibility.

Platform Management Layer

The role of the platform management layer is to provide tools and query languages for accessing NoSQL databases. This layer uses the HDFS storage file system that lies on the top of the Hadoop physical infrastructure layer. The data is no longer stored in a monolithic server where the SQL functions are applied to crunch it. Redundancy is built into this infrastructure for the very simple reason that we are dealing with large volume of data from different sources.



HSN: High-Speed Network

Security Layer

The security layer handles the basic security principles that Big Data architecture should follows. Following security checks must be considered while designing a Big Data stack:

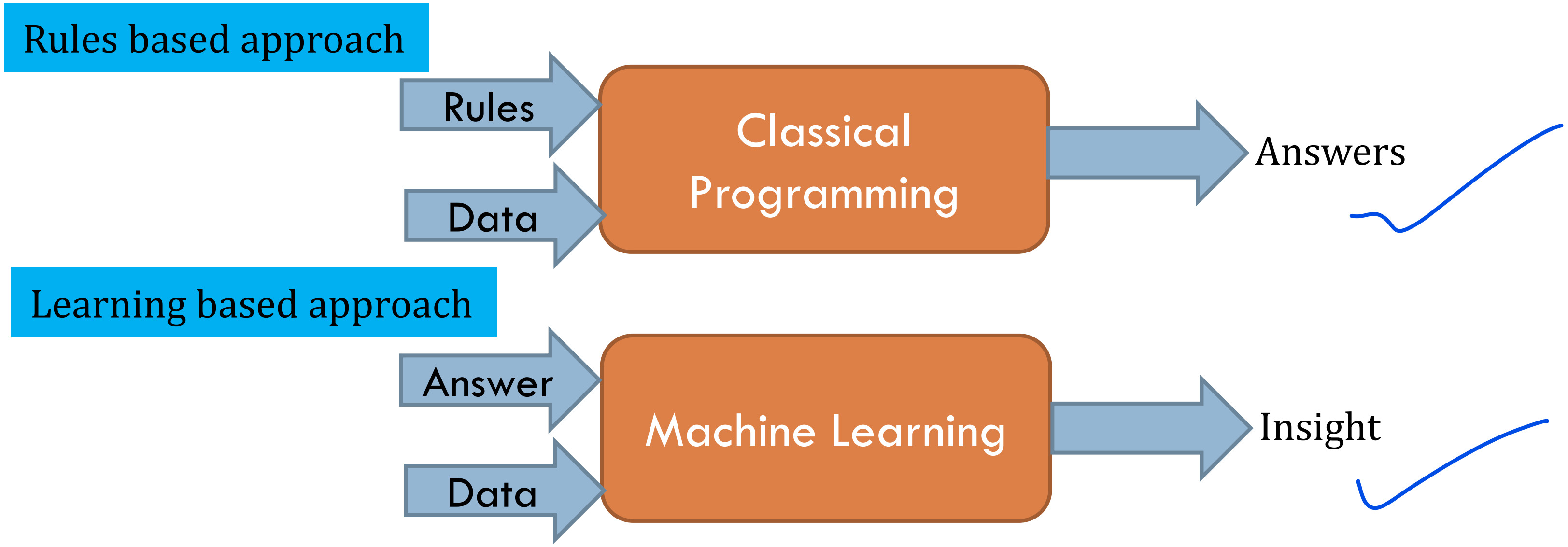
- ❑ It must authenticate nodes by using protocols such as Kerberos
- ❑ It must enable file-layer encryption
- ❑ It must subscribe a key management service for trusted keys and certificates
- ❑ It must maintain logs of the communication that occurs between nodes and trace any anomalies across layers by using distributed logging mechanisms.
- ❑ It must ensure a secure communication between nodes by using the Secure Sockets Layer (SSL)
- ❑ It must validate data during the deployments of datasets or while applying service patches on virtual nodes

Monitoring Layer

The monitoring layer consists of a number of monitoring systems. The system remains automatically aware of all the configurations and functions of different operating systems and hardware. It also provide the facility of machine communication with the help of a monitoring tool through high-level protocols, such as Extensible Markup Language (XML). Monitoring systems also provide tools for data storage and visualization.

Analytics Engine

The role of an analytics engine is to analyze huge amounts of unstructured data. This type of analysis is related to text analytics and statistical analytics. Analytics is broadly the bridge between data and insight. At the outset, the need is to understand (or describe) the data, which is usually done using rule based approach. After which to predict what can happen in order to get some insight and make good decisions usually done with learning based approach.



Note: Contributed by Manoj Kumar Lenka (Roll No: 1605456, 3rd year KIIT CSE student)

Analytics Engine cont’d

Rules based approach

Rules are defined based on which output is generated from the data. It is useful to extract the structural/syntactic meaning of the data. Examples are:

- Finding frequency of words.
- Find particular words in sentences
- Check if a set of words follows a particular pattern, etc..
- Length of sentences.

Learning based approach

The rules are not explicitly programmed and the data is self-learnt. It has two phases namely **training phase** and the **inference (testing) phase**. In the training phase several features (what we have) and their corresponding labels (what to predict) are given to the system, based on which the system learns the relation between them by training its parameters. In the inference phase the trained system is given just the features and using the learned parameters it predicts the output. Learning based systems mainly consists of Machine Learning (ML) models. Examples are:

- Image/Speech Reorganization
- Medical diagnosis
- Recommendation Engine
- Prediction, Extraction

Note: Contributed by Manoj Kumar Lenka (Roll No: 1605456, 3rd year KIIT CSE student)

Analytics Engine cont'd

Some statistical and numerical methods used for analyzing various unstructured data sources are:

- ❑ Natural Language Processing
- ❑ Text Mining
- ❑ Machine Learning
- ❑ Linguistic Computation
- ❑ Search and Sort Algorithms
- ❑ Syntax and Lexical Analysis

Some examples of different types of unstructured data that are available as large dataset include the following:

- ❑ Machine generated data such as RFID feeds and weather data
- ❑ Documents containing textual patterns
- ❑ Data generated from application logs about upcoming or down time details or about maintenance and upgrade details

Analytics Engine cont’d

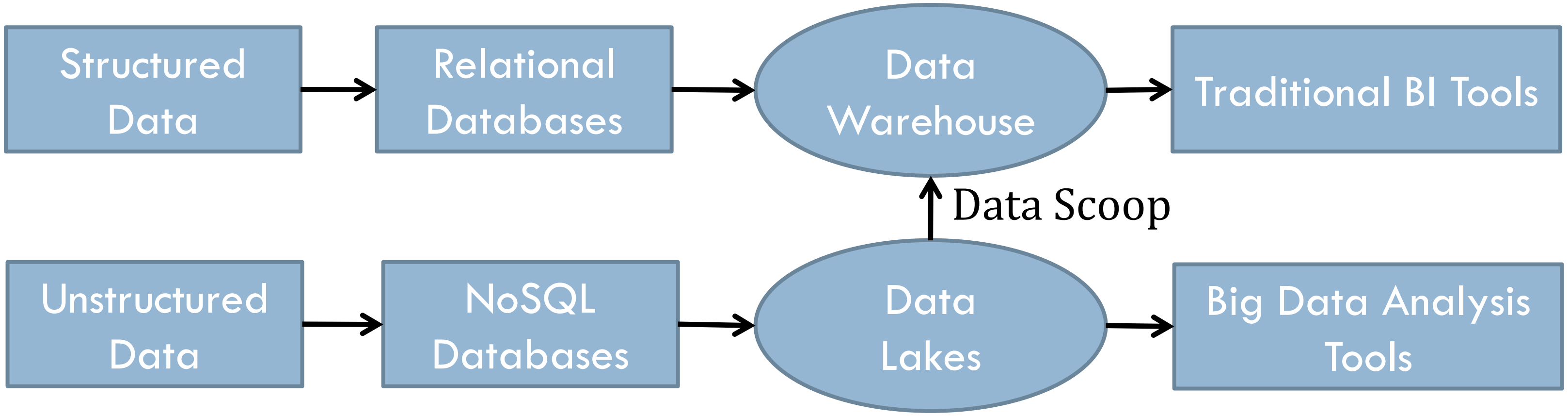
The following types of engines are used for analyzing Big Data:

- **Search engines:** Big data analysis requires extremely fast search engines with iterative and cognitive data discovery mechanisms for analyzing huge volumes of data. This is required because the data loaded from various sources has to be indexed and searched for Big Data analytics processing.
- **Real-time engines**: real-time application in modern era generating data at a very high speed and even a few-hour old data becomes obsolete and useless as new data continues to flow in. Real-time analysis is required [redacted] in the Big Data environment to analyze this type of data. For this purpose, real-type engines and NoSQL data stores are used.

Visualization Layer

It handles the task of interpreting and visualizing Big Data. Visualization of data is done by data analysts and scientists to have a look at the different aspects of the data in various visual modes. It can be described as viewing a piece of information from different perspectives, interpreting it in different manners, trying to fit in different types of situations and deriving different types of conclusions from it.

Some examples of visualization tools are Tableau, Clickview, Spotfire, MapR, and revolution R. These tools work on the top of the traditional components such as reports, dashboards, scorecards, and queries.



Visualization Layer cont'd

A data lake is a centralized repository that allows to store all your structured and unstructured data at any scale. It can store data as-is, without having to first structure the data, and run different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decisions.

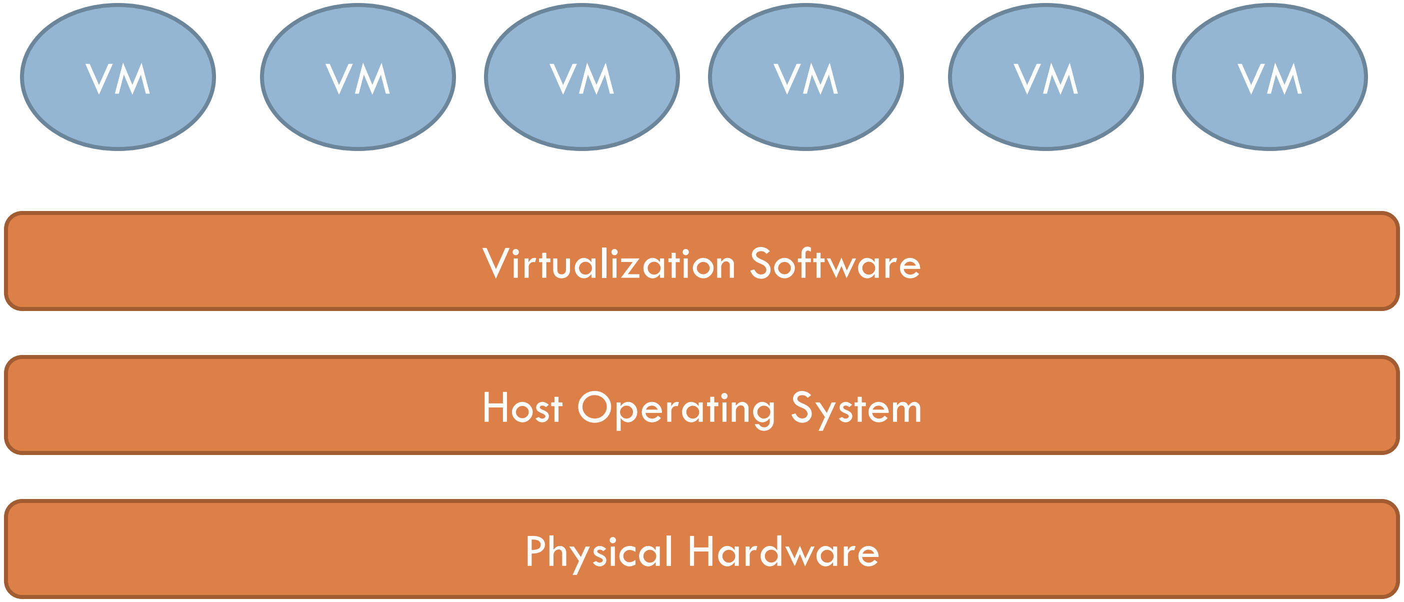
Characteristics	Data Warehouse	Data Lake
Data	Relational from transactional systems, operational databases, and line of business applications	Non-relational and relational from IoT devices, web sites, mobile apps, social media, and corporate applications
Schema	schema-on-write	schema-on-read
Users	Business analysts	Data scientists, and Business analysts
Analytics	Batch reporting, BI and visualizations	Machine Learning, Predictive analytics, and data discovery

Virtualization

In computing, virtualization refers to the act of creating a virtual (rather than actual) version of something, such as virtual computer hardware platforms, storage devices, operating systems and computer network resources. In other words, it is a process to run the images of multiple operating systems on a physical computer. The images of the operating systems are called virtual machines (**VMs**). A virtual machine (**VM**) is basically a software representation of a physical machine that can execute or perform the same functions as the physical machines. Each virtual machine contains a separate copy of the operating systems with its own virtual hardware resources, device drivers, services, and applications. **Although virtualization is not a requirement for Big Data analysis, but works efficiently in a virtualized environment.**

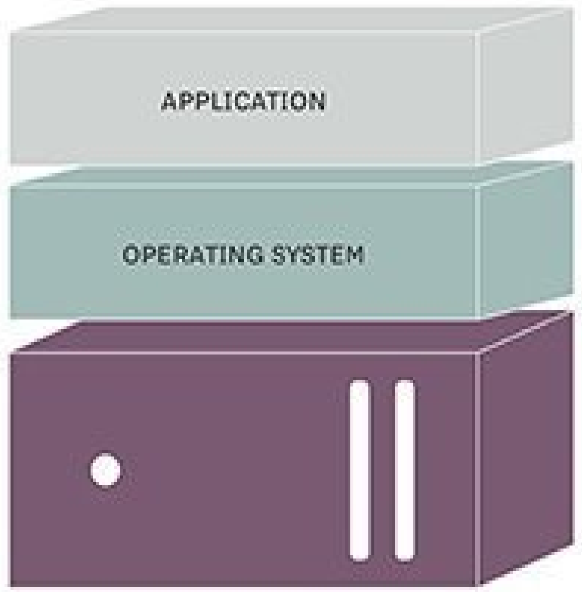
For example, server virtualization is the process in which multiple operating systems (OS) and applications run on the same server at the same time, as opposed to one server running one operating system. If that still seems confusing, think of it as one large server being cut into pieces. The server then imitates or pretends to be multiple servers on the network when in reality it's only one. This offers companies the capability to utilize their resources efficiently and lowers the overall costs that come with maintaining servers.

Virtualization Environment

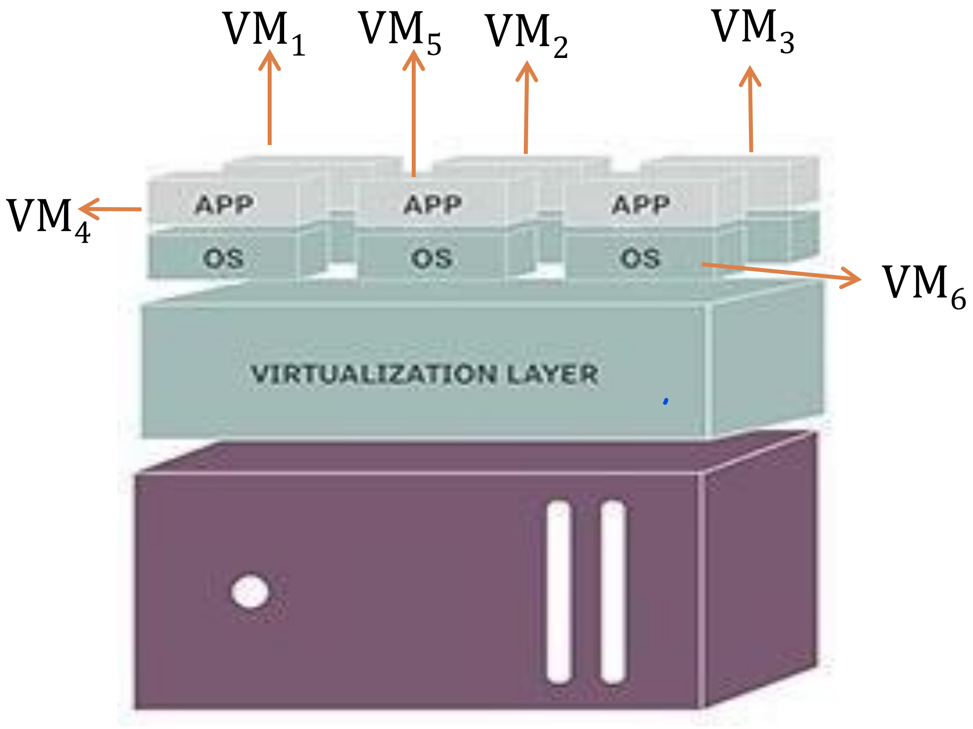


The operating system that runs as a virtual machine is known as the guest, while the operating system that runs as the virtual machine is known as the host. A guest operating system runs on a hardware virtualization layer, which is at the top of the hardware of a physical machine.

Traditional vs. Virtualization System



Traditional System



Virtualization System

Source: researchgate.net

PIEI

Basic Features of Virtualization

- Partitioning: Multiple applications and operating systems are supported by a single physical system by partitioning the available resources.
- Isolation: Each VM runs in an isolated manner from its host physical system and other VMs. If one VM crashes, the other VMs and the host system are not affected.
- Encapsulation: Each VM encapsulates its state as a file system i.e. it can be copied or moved like a simple file.
- Interposition: Generally in a VM, all the new guest actions are performed through the monitor. A monitor can inspect, modify or deny operations such as compression, encryption, profiling, and translation. Such types of actions are done without the knowledge of the operating system.

Types of Virtualization

Virtualization can be utilized in many different ways and can take many forms aside from just server virtualization. The main types include application, desktop, user, storage and hardware.

- ❑ Server virtualization: a single physical server is partitioned into multiple virtual server. Each virtual server has its own hardware and related resources, such as RAM, CPU, hard drive and network controllers. A thin layer of software is also inserted with the hardware which consists of a VM monitor, also called hypervisor and it's to manage the traffic between VMs and the physical machine.
- ❑ Application virtualization allows the user to access the application, not from their workstation, but from a remotely located server. The server stores all personal information and other characteristics of the application, but can still run on a local workstation. Technically, the application is not installed, but acts like it is.
- ❑ Data and Storage virtualization is the process of grouping the physical storage from multiple network storage devices so that it acts as if it's on one storage device.

Types of Virtualization cont'd

- □ Network virtualization: it means using virtual networking as a pool of connection resources. During implementing such virtualization, physical network is to be relied for managing traffic between connections. As many as number of virtual networks can be created from a single physical implementation.
- □ Processor and Memory virtualization: it decouples memory from the server and optimize the power of the processor and maximizes its performance. Big data analysis needs systems to have high processing power (CPU) and memory (RAM) for performing complex computations. These computations can take a lot of time in case CPU and memory resources are not sufficient. Processor and Memory virtualization thus can increase the speed of processing and the analysis results sooner.

Benefits of Virtualization

Virtualization provides several benefits for companies, including:

- ❑ Greater efficiency and company agility
- ❑ Ability to more-effectively manage resources
- ❑ Increased productivity, as employees access the company network from any location
- ❑ Data stored on one centralized server results in a decrease in risk of lost or stolen data

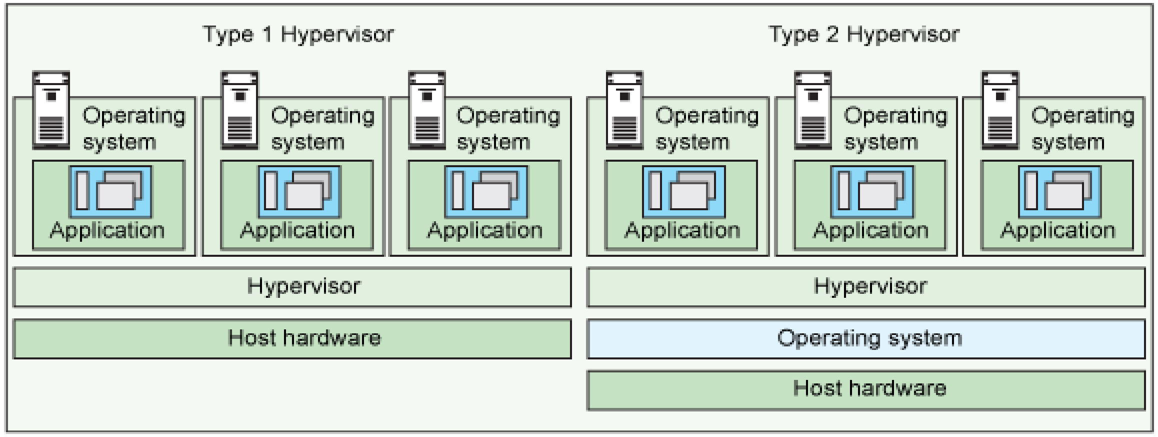
Not only is it beneficial for companies, but virtualization provides several benefits for data centers as well, including:

- ❑ Cutting waste and costs associated with maintaining and cooling its servers by maximizing the capabilities of one server
- ❑ Allows data centers to be smaller in size, resulting in overall savings due to a reduction in:
  - ❑ Energy needed
  - ❑ Hardware used
  - ❑ Time and money needed for maintenance

Managing Virtualization with Hypervisor

Hypervisor is a program that allows multiple operating systems to share a single hardware. It is also known as virtual machine manager. It controls the host processor & resources and allocates the guest operating systems don't disrupt each other. There are 2 types of hypervisor

1. **Type 1**: It runs directly on the system hardware with VM resources provided by the hypervisor. It is often called native or embedded hypervisors. Example are VMwareESXi, Oracle VM and Citrix XenServer.
2. **Type 2**: It runs on a host operating system to provide the virtualization services. Admin can buy the software and install on a server. It is often called as hosted hypervisor. Example are Vmware Server, Sun VirtualBox, and Microsoft Virtual PC.



Source: vapour-apps.com

Data Stream

Data Stream – Large data volume, likely unstructured and structured arriving at a very high rate, which requires real time/near real time analysis for effective decision making.

- It is basically continuously generated data and arrives in a stream (sequence of data elements made available over time). It is generally time-stamped and geo-tagged (in the form of latitude and longitude)
- Stream is composed of synchronized sequence of elements or events
- If it is not processed immediately, then it is lost forever
- In general, such data is generated as part of application logs, events, or collected from a large pool of devices continuously generating events such as ATM or PoS

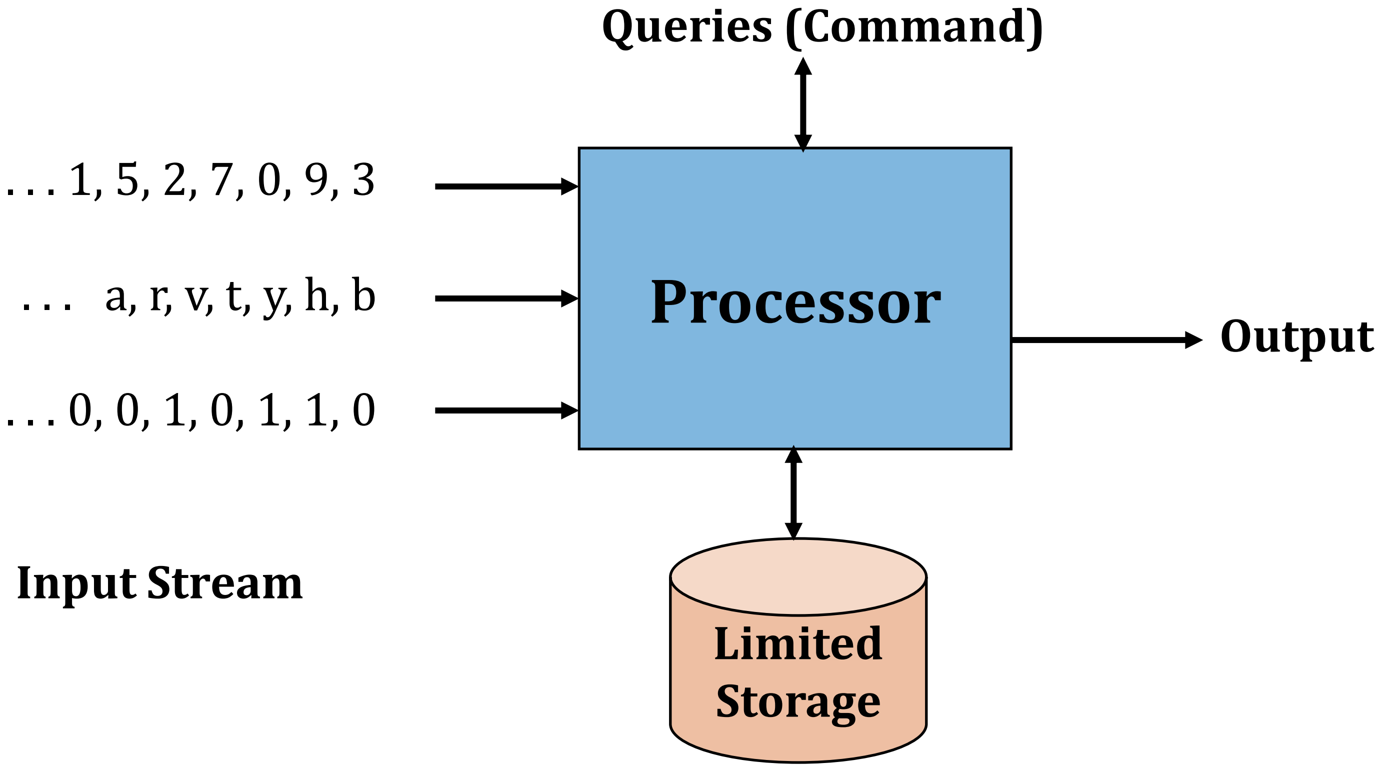
Example:

Data Center: Large network deployment of a data center with hundreds of servers, switches, routers and other devices in the network. The event logs from all these devices at real time create a stream of data. This data can be used to prevent failures in the data center and automate triggers so that the complete data center is fault tolerant

Stock Market: The data generated here is a stream of data where a lot of events are happening in real-time. The price of stock are continuously varying. These are large continuous data streams which needs analysis in real-time for better decisions on trading

Basic Model of Stream data

- Input data rapidly and streams needn't have the same data rates or data types
- The system cannot store the data entirely
- Queries tends to ask information about recent data
- The scan never turn back



Big Data Streaming

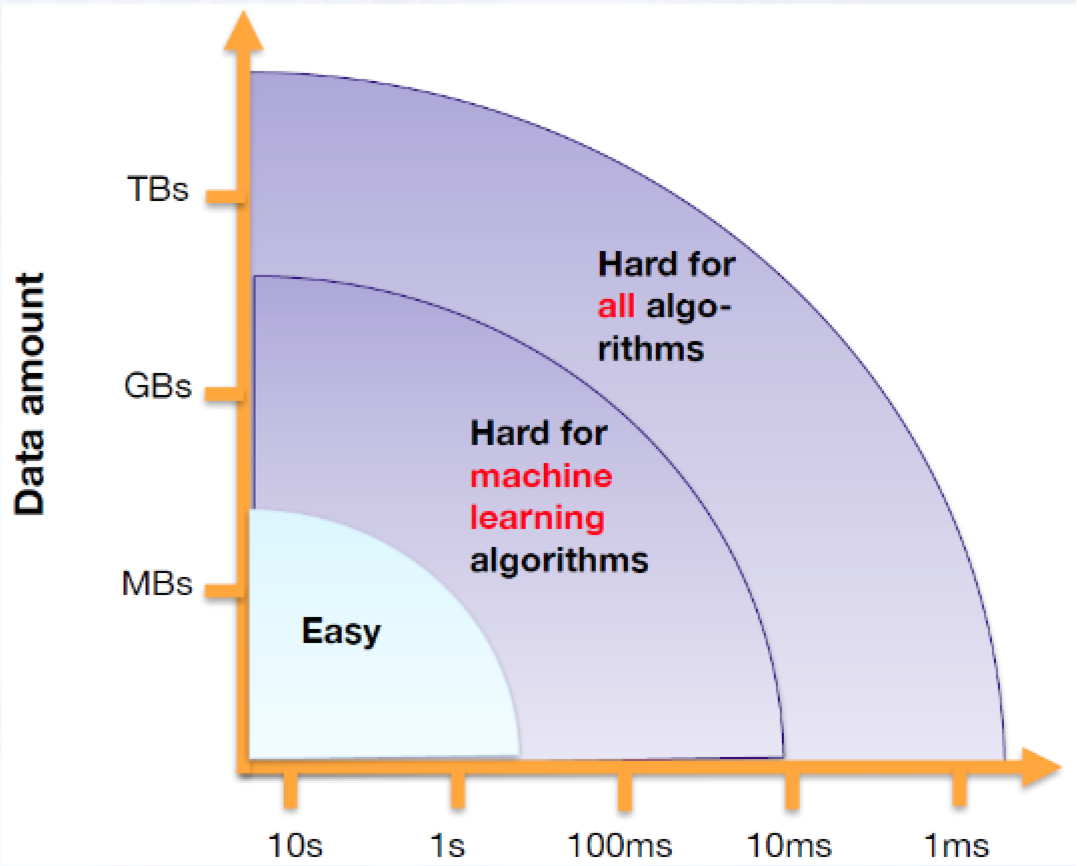
Big data streaming is a process in which big data is quickly processed in order to extract real-time insights from it. The data on which processing is done is the data in motion. Big data streaming is ideally a speed-focused approach wherein a continuous stream of data is processed.

For real-time big data stream processing, following three key attributes are required:

- ❑ System to collect the big data generated in real time
- ❑ System to handle the massive parallel processing of this data
- ❑ Event correlation and event processing engine for generating analytics

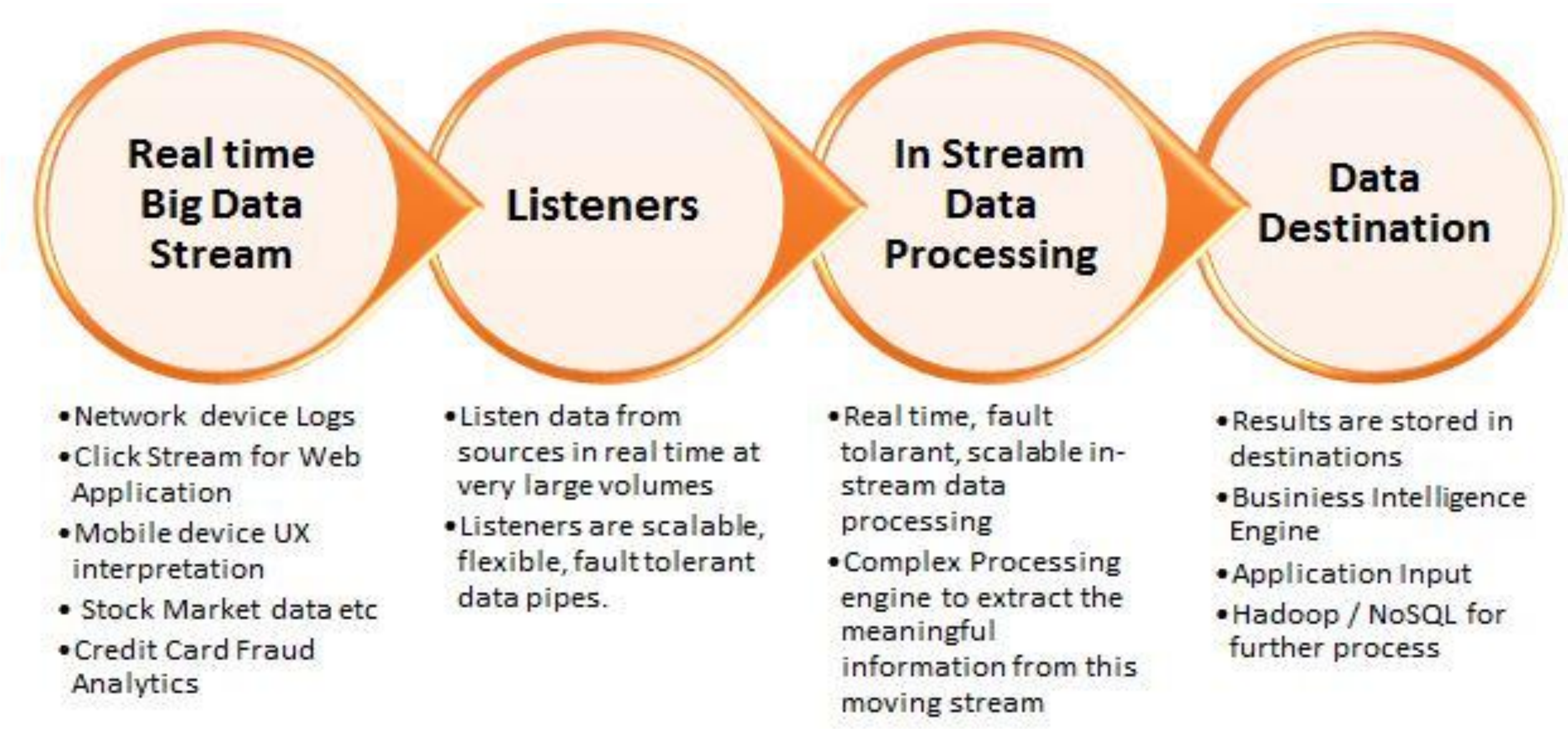
All the above mentioned attributes / components need to be fault tolerant, scalable and distributed, with low latency for each system.

Why Big Data Streaming is hard?



Source: Tilmann Rabl, Big Data Stream Processing

Big Data Streaming Process



Real time Big Data Stream

- Network device Logs
- Click Stream for Web Application
- Mobile device UX interpretation
- Stock Market data etc
- Credit Card Fraud Analytics

Listeners

- Listen data from sources in real time at very large volumes
- Listeners are scalable, flexible, fault tolerant data pipes.

In Stream Data Processing

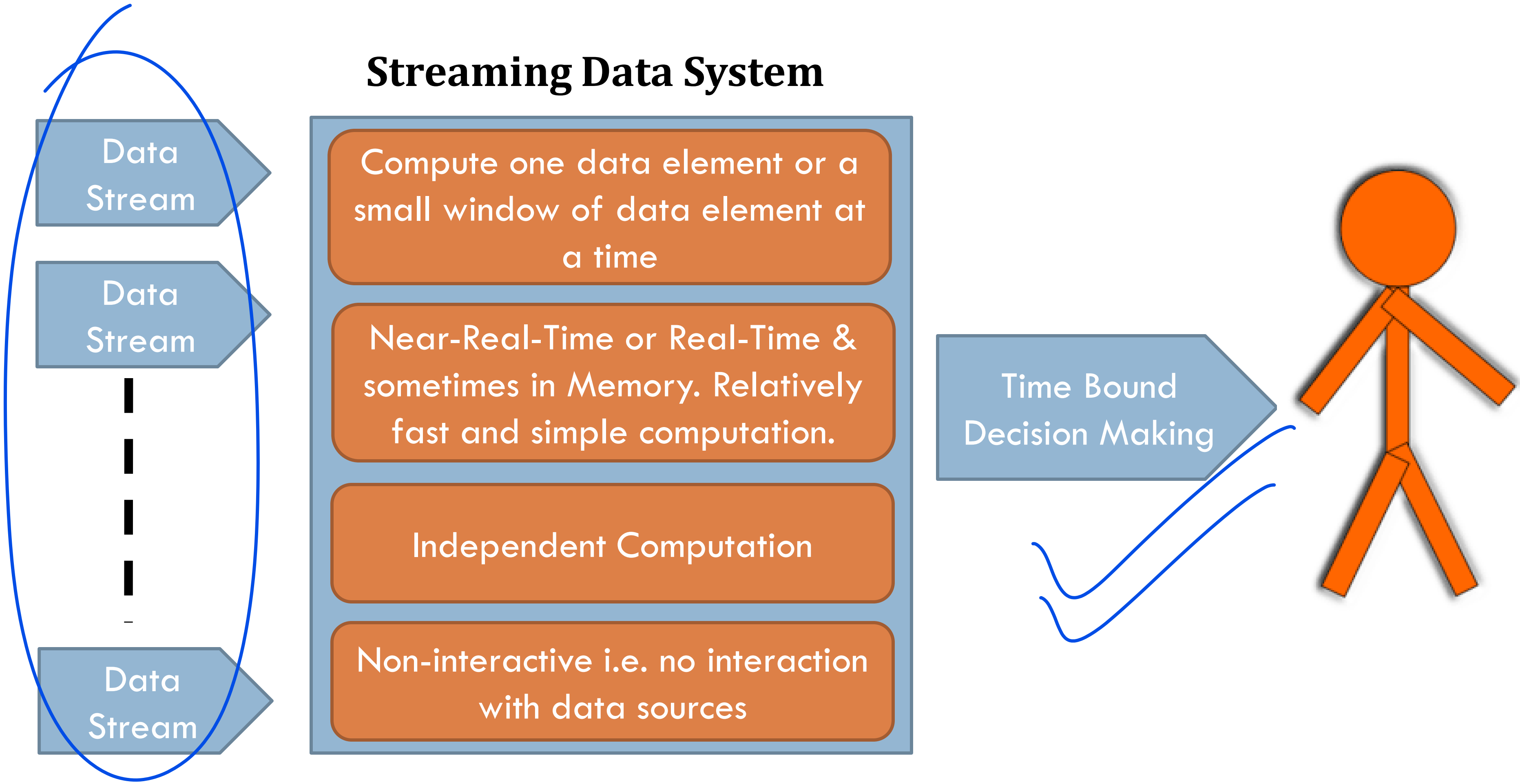
- Real time, fault tolarant, scalable in-stream data processing
- Complex Processing engine to extract the meaningful information from this moving stream

Data Destination

- Results are stored in destinations
- Businiess Intelligence Engine
- Application Input
- Hadoop / NoSQL for further process

Source: https://blog.blazeclan.com/

Streaming Data System



Streaming Data System
Data Stream
Data Stream
Data Stream
Compute one data element or a small window of data element at a time
Near-Real-Time or Real-Time & sometimes in Memory. Relatively fast and simple computation.
Independent Computation
Non-interactive i.e. no interaction with data sources
Time Bound Decision Making

Data-at-Rest vs. Data-in-Motion

- Data-at-rest - This refers to data that has been collected from various sources and is then analyzed after the event occurs. The point where the data is analyzed and the point where action is taken on it occur at two separate times. For example, a retailer analyzes a previous month's sales data and uses it to make strategic decisions about the present month's business activities. The action takes place after the data-creating event has occurred. For data at rest, a batch processing method would be most likely.
- Data-in-motion - The collection process for data in motion is similar to that of data at rest; however, the difference lies in the analytics. In this case, the analytics occur in real-time as the event happens. For example – sensor data from self-driving vehicles. For data in motion, you'd want to utilize a real-time processing method.

Data-at-Rest vs. Data-in-Motion Infrastructure Option

Data-at-rest

Public Cloud

Public cloud can be an ideal infrastructure choice in such scenario from a cost standpoint, since virtual machines can easily be spun up as needed to analyze the data and spun down when finished.

Data-in-motion

Bare-Metal Cloud

Bare-Metal cloud can be an preferable infrastructure choice. It involves the use of dedicated servers that offers cloud-like features without the use of virtualization.

Streaming Data Changes over Time

Change can be periodic or sporadic

Periodic: evening, weekends etc

People post Facebook messages more in the evening in comparison to during day, working hours.

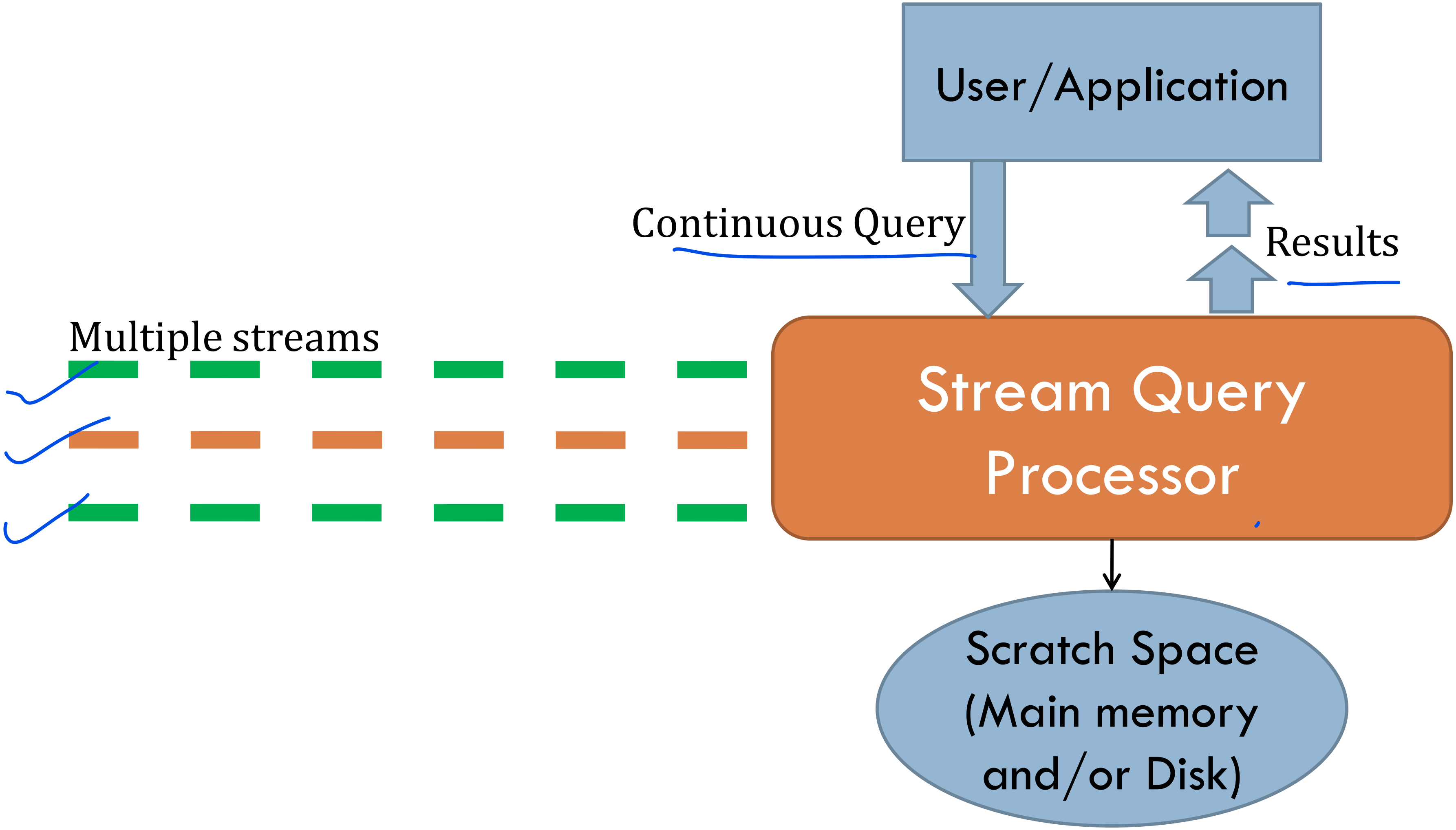
Sporadic: major events

BREAKING NEWS

In summary, streaming data:

- ❑ Size is unbounded i.e. it continually generated and can’t process all at once
- ❑ Size and Frequency is Unpredictable due to human behavior
- ❑ Processing is must be relatively fast and simple

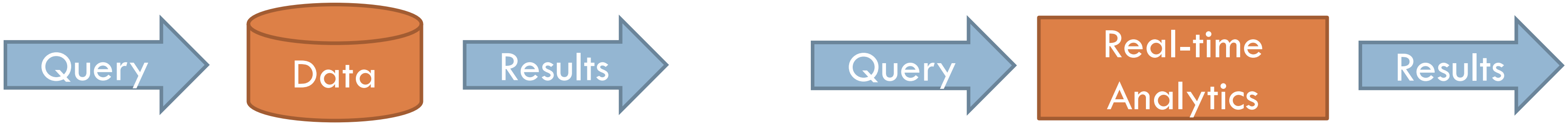
Architecture: Stream Query Processing



Stream Computing

A high-performance computer system that analyzes multiple data streams from many sources. The word stream in stream computing is used to mean pulling in streams of data, processing the data and streaming it back out as a single flow. Stream computing uses software algorithms that analyzes the data in real time as it streams in to increase speed and accuracy when dealing with data handling and analysis.

Traditional Computing	Stream Computing
Historical fact finding	Current fact finding
Find and analyze information stored on disk	Analyze data in motion – before it is stored
Batch paradigm, pull model	Low latency paradigm, push model
Query-driven – queries data to static data	Data-driven – bring data to the analytics



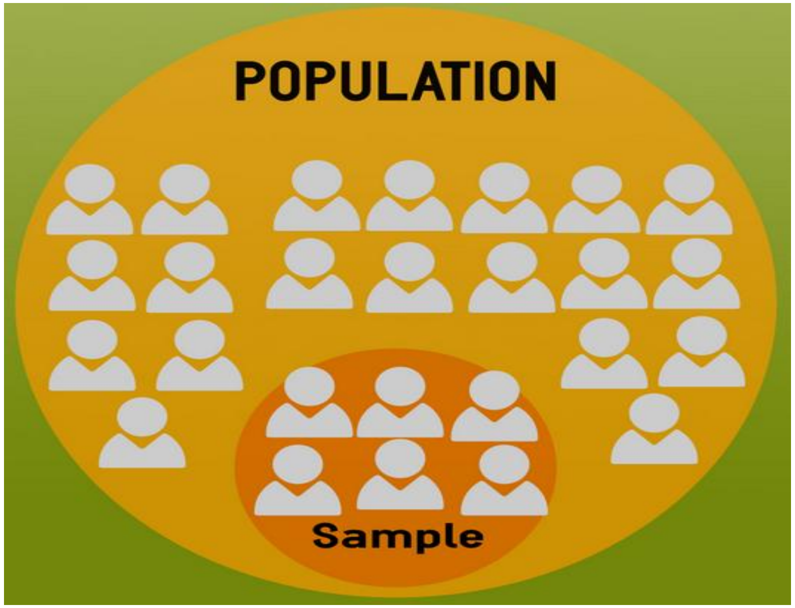
Query Types on Data Stream

Types of queries one wants on answer on a data stream:

- ❑ Sampling data from a stream - construct a random sample
- ❑ Queries over sliding windows - number of items of type x in the last k elements of the stream
- ❑ Filtering a data stream - select elements with property x from the stream
- ❑ **Counting distinct elements** - number of distinct elements in the last k elements of the stream or in the entire stream
- ❑ Estimating moments - estimate avg./std. dev. of last k elements
- ❑ Finding frequent elements

Data Sampling

Data sampling is a technique used to select, manipulate and analyze a representative subset of data points to identify patterns and trends in the larger data set being examined. It enables data scientists, predictive modelers and other data analysts to work with a small, manageable amount of data to build and run analytical models more quickly, while still producing accurate findings.



Source: Google Analytics Data Sampling

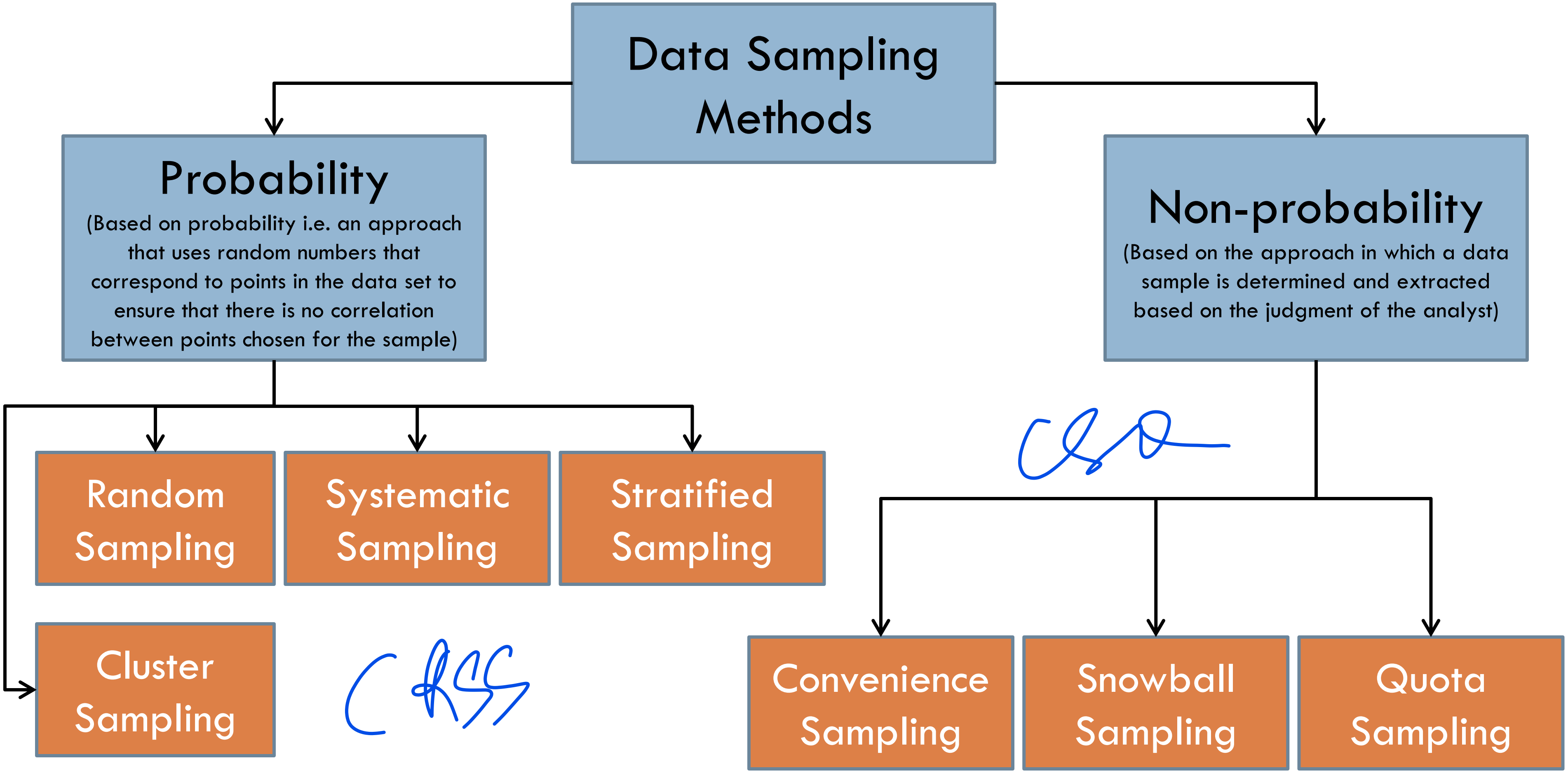
Why Sampling is Important?

- **Resource Constraints:** If the target population is not small enough, or if the resources at the disposal don't give the bandwidth to cover the entire population, it is important to identify a subset of the population to work with – a carefully identified group that is representative of the population. This process is called survey sampling. Whatever the sample size, there are fixed costs associated with any survey. Once the survey has begun, the marginal costs associated with gathering more information, from more people, are proportional to the size of the sample.
- **Drawing Inferences About the Population**: Researcher are not interested in the sample itself, but in the understanding that they can potentially infer from the sample and then apply across the entire population. Working within a given resource constraint, sampling may make it possible to study the population of a larger geographical area or to find out more about the same population by examining an area in greater depth through a smaller sample.

Features to Keep in mind while Constructing a Sample

- **Consistency:** It is important that researchers understand the population on a case-by-case basis and test the sample for consistency before going ahead with the survey.
- **Diversity**: A sample must be as diverse as the population itself and sensitive to the differences that are unavoidable as we move across the population.
- **Transparency:** There are several constraints that dictate the size and structure of the population. It is imperative that researchers discuss these limitations and maintain transparency about the procedures followed while selecting the sample so that the results of the survey are seen with the right perspective.

Data Sampling Methods


Data Sampling Methods
Probability
(Based on probability i.e. an approach that uses random numbers that correspond to points in the data set to ensure that there is no correlation between points chosen for the sample)
Non-probability
(Based on the approach in which a data sample is determined and extracted based on the judgment of the analyst)
Random Sampling
Systematic Sampling
Stratified Sampling
Cluster Sampling
CRSS
CSQ
Convenience Sampling
Snowball Sampling
Quota Sampling

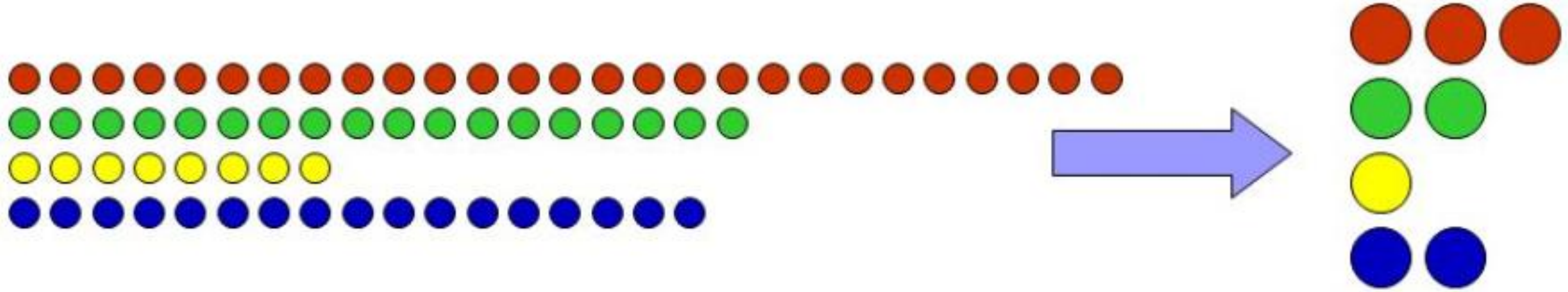
Data Sampling Methods cont’d

- **Random Sampling:** Every item in the population has an even chance and likelihood of being selected in the sample. Here the selection of items completely depends on chance or by probability and therefore this sampling technique is also sometimes known as a **method of chances.**
- **Systematic Sampling**: Elements are chosen from a target population by selecting a random starting point and selecting other members after a fixed ‘sampling interval’ and it is calculated by dividing the entire population size by the desired sample size.
- **Stratified Sampling:** Entire population is divided into different subgroups or strata based on common factor, then randomly selects the final subjects proportionally from the different strata.
- **Cluster Sampling**: The larger data set is divided into subsets or clusters based on a defined factor, then a random sampling of clusters is analyzed.

Data Sampling Methods cont’d

- **Convenience Sampling:** The elements of such a sample are picked only on the basis of convenience in terms of availability, reach and accessibility.
- **Snowball Sampling**: Just as the snowball rolls and gathers mass, the sample constructed in this way will grow in size as you move through the process of conducting a survey. In this technique, you rely on your initial respondents to refer you to the next respondents whom you may connect with for the purpose of your survey.
- **Quota Sampling:** Survey population is divided into mutually exclusive subgroups. These subgroups are selected with respect to certain known features, traits, or interests. People in each subgroup are selected by the researcher or interviewer who is conducting the survey.

Sampling data in Stream



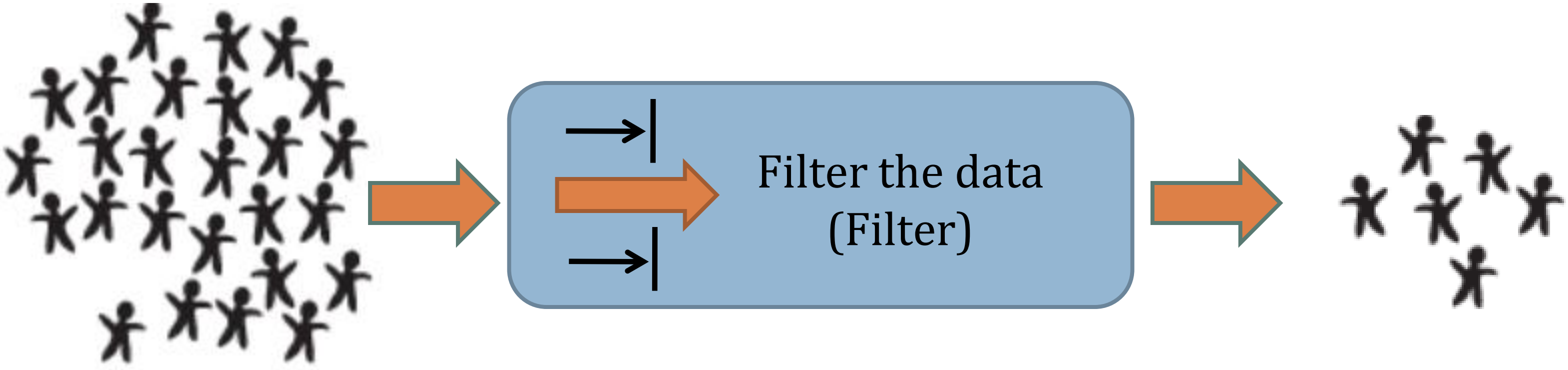
Need & how of sampling - System cannot store the entire stream conveniently, so

- how to make critical calculations about the stream using a limited amount of (primary or secondary) memory?
- Don’t know how long the stream is, so when and how often to sample?

Whenever a list or set is used, and space is consideration, a **Bloom filter** should be considered.

Filter

A filter is a is a program or section of code that is designed to examine each input or output stream for certain qualifying criteria and then process or forward it accordingly by producing another stream.



All stock transaction
(Input)

KIIT stock transaction
(Output)

In this example, the streams processing application needs to filter the stock transaction data for KIIT transaction records.

Bloom Filter

Bloom filter is a space-efficient probabilistic data structure conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set. **False positive** matches are possible, but **False Negatives are not** – in other words, a query returns either "**possibly in set**" or "**definitely not in set**"

False Positive = "**possibly in set**" or "**definitely not in set**"

False Negative = "**possibly not in set**" or "**definitely in set**"

Overview

x : An element

S: A set of elements

Input: x, S Output:

-TRUE if x in S

-FALSE if x not in S

Bloom Filter cont’d

Suppose you are creating an account on Facebook, you want to enter a cool username, you entered it and got a message, “Username is already taken”. You added your birth date along username, still no luck. Now you have added your university roll number also, still got “Username is already taken”. It’s really frustrating, isn’t it?

But have you ever thought how quickly Facebook check availability of username by searching millions of username registered with it. There are many ways to do this job –

Linear search : Bad idea!

Binary Search : There must be something better!!

Bloom Filter is a data structure that can do this job.

Bloom Filter cont'd

For understanding bloom filters, one must know hashing. A hash function takes input and outputs a unique identifier of fixed length which is used for identification of input.

Properties of Bloom Filter

- Unlike a standard hash table, a Bloom filter of a fixed size can represent a set with an arbitrarily large number of elements.
- Adding an element never fails. However, the false positive rate increases steadily as elements are added until all bits in the filter are set to 1, at which point all queries yield a positive result.
- Bloom filters never generate false negative result, i.e., telling you that a username doesn't exist when it actually exists.
- Deleting elements from filter is not possible because, if we delete a single element by clearing bits at indices generated by k hash functions, it might cause deletion of few other elements.

Working of Bloom Filter

A empty bloom filter is a bit array of n bits, all set to zero, like below:

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

We need k number of hash functions to calculate the hashes for a given input. When we want to add an item in the filter, the bits at k indices $h_1(x)$, $h_2(x)$, … $h_k(x)$ are set, where indices are calculated using hash functions.

5

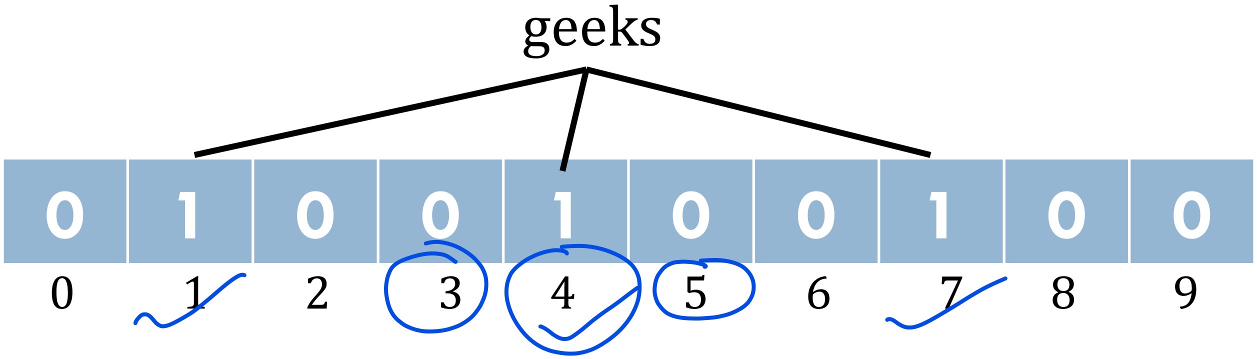
Example – Suppose we want to enter "geeks" in the filter, we are using 3 hash functions and a bit array of length 10, all set to 0 initially. First we'll calculate the hashes as following :

h_1("geeks") % 10 = 1, h_2("geeks") % 10 = 4, and h_3("geeks") % 10 = 7

Note: *These outputs are random for explanation only.*

Working of Bloom Filter cont’d

Now we will set the bits at indices 1, 4 and 7 to 1



Again we want to enter “nerd”, similarly we’ll calculate hashes

h_1(“nerd”) % 10 = 3

h_2(“nerd”) % 10 = 5

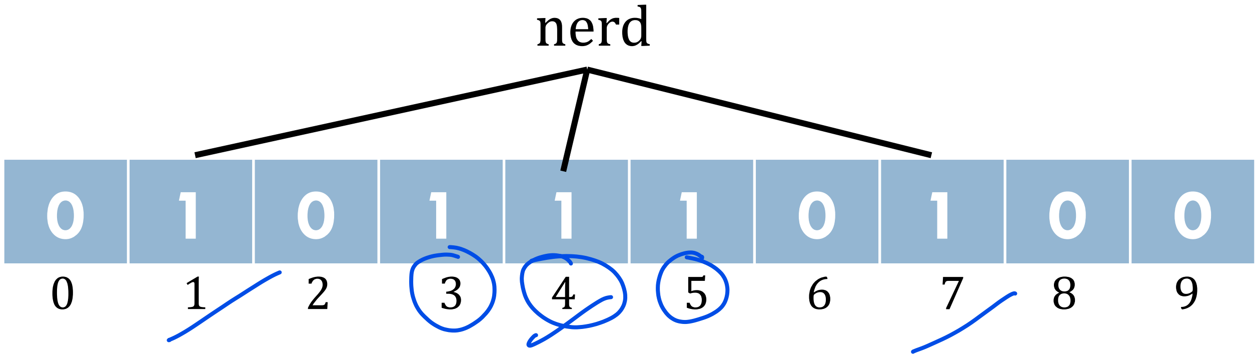
h_3(“nerd”) % 10 = 4

Note: *These outputs are random for explanation only.*

Set the bits at indices 3, 5 and 4 to 1

Working of Bloom Filter cont’d

Now we will set the bits at indices 3, 5 and 4 to 1



0	1	0	1	1	1	0	1	0	0
0	1	2	3	4	5	6	7	8	9

Now if we want to check “geeks” is present in filter or not. We’ll do the same process but this time in reverse order. We calculate respective hashes using h_1, h_2 and h_3 and check if all these indices are set to 1 in the bit array. If all the bits are set then we can say that “geeks” is probably present. If any of the bit at these indices are 0 then “geeks” is definitely not present.

Bloom Filter Algorithm

Insertion

Data: e is the element to insert into the Bloom filter.

```
insert(e)
begin
 /* Loop all hash functions k */
 for j : 1 . . . k do
   m ← hj(e) //apply the hash function on e
   Bm ← bf[m] //retrive val at mth pos from Bloom filter bf
   if Bm == 0 then
      /* Bloom filter had zero bit at index m */
      Bm ← 1;
   end if
 end for
end
```

Lookup

Data: x is the element for which membership is tested.

```
bool isMember(x) /* returns true or false to the membership test */
begin
 t ← 1
 j ← 1
  while t == 1 and j ≤ k do
   m ← hj(x)
   Bm ← bf[m]
    if Bm == 0 then
      t ← 0
    end
    j ← j + 1;
   end while
 return (bool) t
end
```

Bloom Filter Performance

A Bloom filter requires space O(n) and can answer membership queries in O(1) time where n is number item inserted in the filter. Although the asymptotic space complexity of a Bloom filter is the same as a hash map, O(n), a Bloom filter is more space efficient.

Class Exercise

- A empty bloom filter is of size 11 with 4 hash functions namely
  - $h_1(x) = (3x+ 3) \bmod 6$
  - $h_2(x) = (2x+ 9) \bmod 2$
  - $h_3(x) = (3x+ 7) \bmod 8$
  - $h_4(x) = (2x+ 3) \bmod 5$

  Illustrate bloom filter insertion with 7 and then 8.

  Perform bloom filter lookup/membership test with 10 and 48

False Positive in Bloom Filters

The question is why we said “probably present”, why this uncertainty. Let’s understand this with an example. Suppose we want to check whether “cat” is present or not. We’ll calculate hashes using h_1, h_2 and h_3

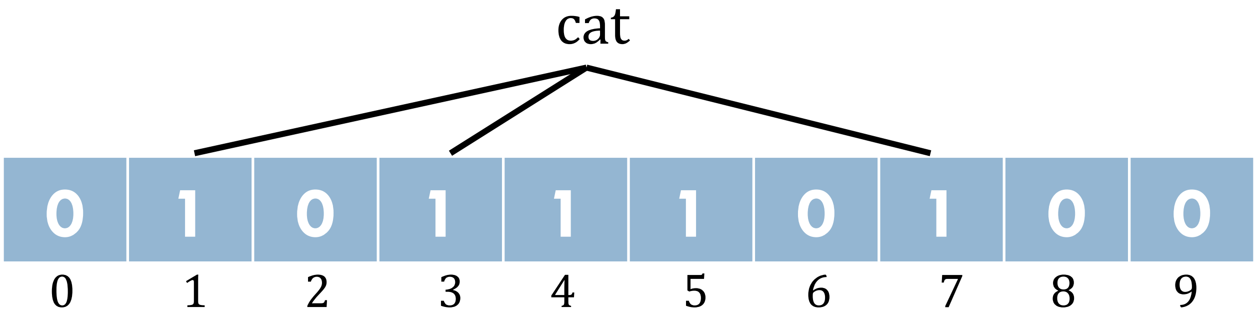
h_1(“cat”) % 10 = 1

h_2(“cat”) % 10 = 3

h_3(“cat”) % 10 = 7

If we check the bit array, bits at these indices are set to 1 but we know that “cat” was never added to the filter. Bit at index 1 and 7 was set when we added “geeks” and bit 3 was set we added “nerd”.

False Positive in Bloom Filters cont’d



0	1	2	3	4	5	6	7	8	9
0	1	0	1	1	1	0	1	0	0

So, because bits at calculated indices are already set by some other item, bloom filter erroneously claim that “cat” is present and generating a false positive result. Depending on the application, it could be huge downside or relatively okay.

We can control the probability of getting a false positive by controlling the size of the Bloom filter. More space means fewer false positives. If we want decrease probability of false positive result, we have to use more number of hash functions and larger bit array. This would add latency in addition of item and checking membership.

False Positive in Bloom Filters cont'd

0	0	0	0	0	0	0	0	0	0	0	0
1	2	3	4	5	6	7	8	9	10	11	12

K=3 (# of Hash Function)

Note: *These outputs are random for explanation only.*

INSERT (x_1)

$h_1(x_1) = 3$

$h_2(x_1) = 5$

$h_3(x_1) = 11$

INSERT (x_2)

$h_1(x_2) = 1$

$h_2(x_2) = 7$

$h_3(x_2) = 6$

State of Hashtable post to the insertion of x_1 and x_2

1	0	1	0	1	1	1	0	0	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11

False Positive in Bloom Filters cont'd

1	0	1	0	1	1	1	0	0	0	1	0
0	1	2	3	4	5	6	7	8	9	10	11

LOOKUP(x_3)

$h_1(x_3) = 3$

$h_2(x_3) = 11$

$h_3(x_3) = 7$

✓

Case of FALSE POSITIVE

LOOKUP(x_4)

$h_1(x_4) = 4$

$h_2(x_4) = 9$

$h_3(x_4) = 10$

✓

X_4 doesn't exist

Optimum number of hash functions

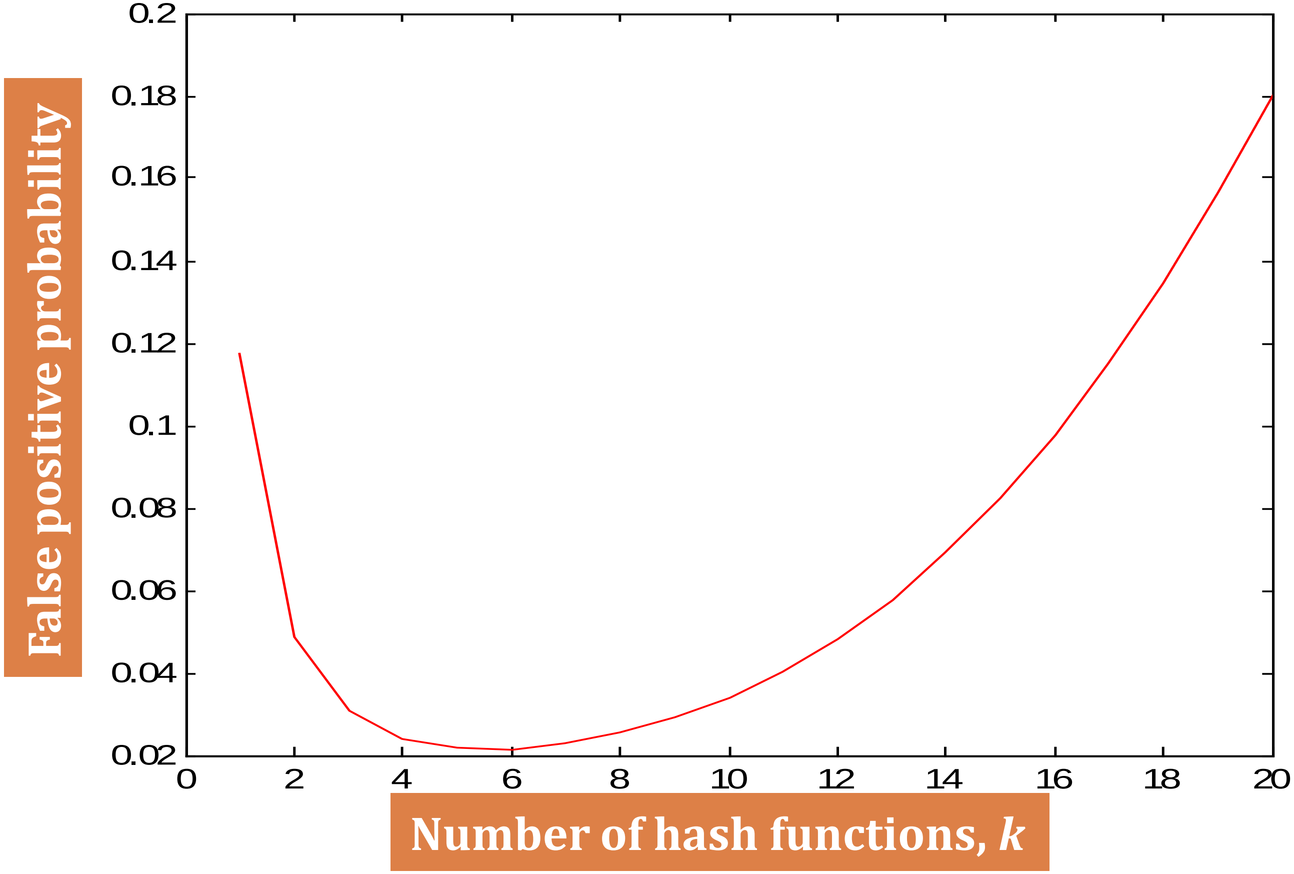
The number of hash functions k must be a positive integer. If n is size of bit array and m is number of elements to be inserted, then k can be calculated as :

$$k = \frac{n}{m} \ln(2)$$

Class Exercise

Calculate the optimal number of hash functions for 10 bit length bloom filter having 3 numbers of input elements.

What happens to increasing k?



Calculating probability of False Positives

- ❑ Probability that a slot is hashed = 1/n
- ❑ Probability that a slot is not hashed = 1 – (1/n)
- ❑ Probability that a slot is not hashed after insertion of an element for all the k hash function is:

$$\left(1-\frac{1}{n}\right)^{k}$$

- ❑ Probability that a slot is not set to 1 after insertion of m element is:

$$\left(1-\frac{1}{n}\right)^{km}$$

- ❑ Probability that a slot is set to 1 after insertion of m element is:

$$1-\left(1-\frac{1}{n}\right)^{km}$$

Calculating probability of False Positives cont’d

Let n be the size of bit array, k be the number of hash functions and m be the number of expected elements to be inserted in the filter, then the probability of false positive p can be calculated as:

$$\left(1-\left(\frac{1}{e}\right)^{\frac{km}{n}}\right)^{k}$$

$$\boxed{e = 2.71}$$

Class Exercise

Calculate the probability of False Positives with table size 10 and no. of items to be inserted are 3.

Counting distinct elements in a stream – Approach 1

(1, 2, 2, 1, 3, 1, 5, 1, 3, 3, 3, 2, 2)

Number of distinct elements = 4

How to calculate?

1. Initialize the hashtable (large binary array) of size n with all zeros.
2. Choose the hash function $h_i : i \in \{1, ..., k\}$
3. For each flow label $f \in \{1, ..., m\}$, compute h(f) and mark that position in the hashtable with 1
4. Count the number of positions in the hashtable with 1 and call it c.
5. The number of distinct items is m* ln (m / (m-c))

Class Exercise

Count the distinct elements in a data stream of elements {1, 2, 2, 1, 3, 1, 5, 1, 3, 3, 3, 2, 2} with the hash function h(x) = (5x+1) mod 6 of size 11.

Counting distinct elements in a stream using Flajolet-Martin algorithm – Approach 2

Count the distinct elements in a data stream of elements {6,8,4,6,3,4} with hash function h(x) = (5x+1) mod 6 of size 11.

How to calculate?

Step 1:

Apply Hash function(s) to the data stream and compute the slots.

h(6)=1, h(8)=5, h(4)=3, h(6)=1, h(3)=4, h(4)=3.

The slot numbers obtained are:{1,5,3,1,4,3}

Step 2: Convert the numbers to binary

h(6)=1=001, h(8)=5=101, h(4)=3=001, h(6)=1=001, h(3)=4=100, h(4)=3=011

Step 3: Calculate the maximum trailing zeros

TZ ={0, 0, 0, 0, 2, 0} /* TZ stands for Trailing Zeros */

R = MAX(TZ) = MAX(0, 0, 0, 0, 2, 0) = 2

Step 4: Estimate the distinct elements with the formula 2^R

Number of distinct elements = $2^R = 2^2 = 4$

Bloom Filter Use Cases

- Bitcoin uses Bloom filters to speed up wallet synchronization and also to improve Bitcoin wallet security
- Google Chrome uses the Bloom filter to identify malicious URLs - it keeps a local Bloom filter as the first check for Spam URL
- Google BigTable and Apache Cassandra use Bloom filters to reduce disk lookups for non-existent rows or columns
- The Squid Web Proxy Cache uses Bloom filters for cache digests - proxies periodically exchange Bloom filters for avoiding ICP messages
- Genomics community uses Bloom filter for classification of DNA sequences and efficient counting of k-mers in DNA sequences
- Used for preventing weak password choices using a dictionary of easily guessable passwords as bloom filter
- Used to implement spell checker using a predefined dictionary with large number of words

Other Types of Bloom Filter

- **Compressed Bloom Filter** - Using a larger but sparser Bloom Filter can yield the same false positive rate with a smaller number of transmitted bits.
- **Scalable Bloom Filter** - A Scalable Bloom Filters consist of two or more Standard Bloom Filters, allowing arbitrary growth of the set being represented.
- **Generalized Bloom Filter** - Generalized Bloom Filter uses hash functions that can set as well as reset bits.
- **Stable Bloom Filter** - This variant of Bloom Filter is particularly useful in data streaming applications.

Develop Flajolet-Martin algorithm and using it, count the distinct elements in a data stream of elements {6, 8, 4, 6, 3, 4, 7, 6, 9} with the hash function h(x) = (5x+1) mod 6 of size 11.

A empty bloom filter is of size 11 with 4 hash functions namely:
h1(x) = (3x+ 3) mod 6
h2(x) = (2x+ 9) mod 2
h3(x) = (3x+ 7) mod 8
h4(x) = (2x+ 3) mod 5
Illustrate bloom filter insertion with 17, 81 and 37.
Perform bloom filter lookup/membership test with 10 and 81.

A empty bloom filter is of size 11 with 2 hash functions namely:
h1(x) = (3x+ 3) mod 18
h2(x) = (2x+ 9) mod 22
Illustrate bloom filter insertion with 7, 8 and 77.
Perform bloom filter lookup/membership test with 7, 10 and 88.

Develop an algorithm to i) insert an item, and to ii) test the membership (or lookup) in Bloom Filter. Draw a step-by-step process in the insertion of element 25, and then 40 into the Bloom Filter of size 10. Then, draw a step-by-step process for lookup/membership test with the elements 10 and 48. The hash functions are: h1(x) = (3x+41) mod 6, and h2(x) = (7x+5). Identify whether any lookup element (i.e. either 10 or 48) is resulting into the case of FALSE POSITIVE?