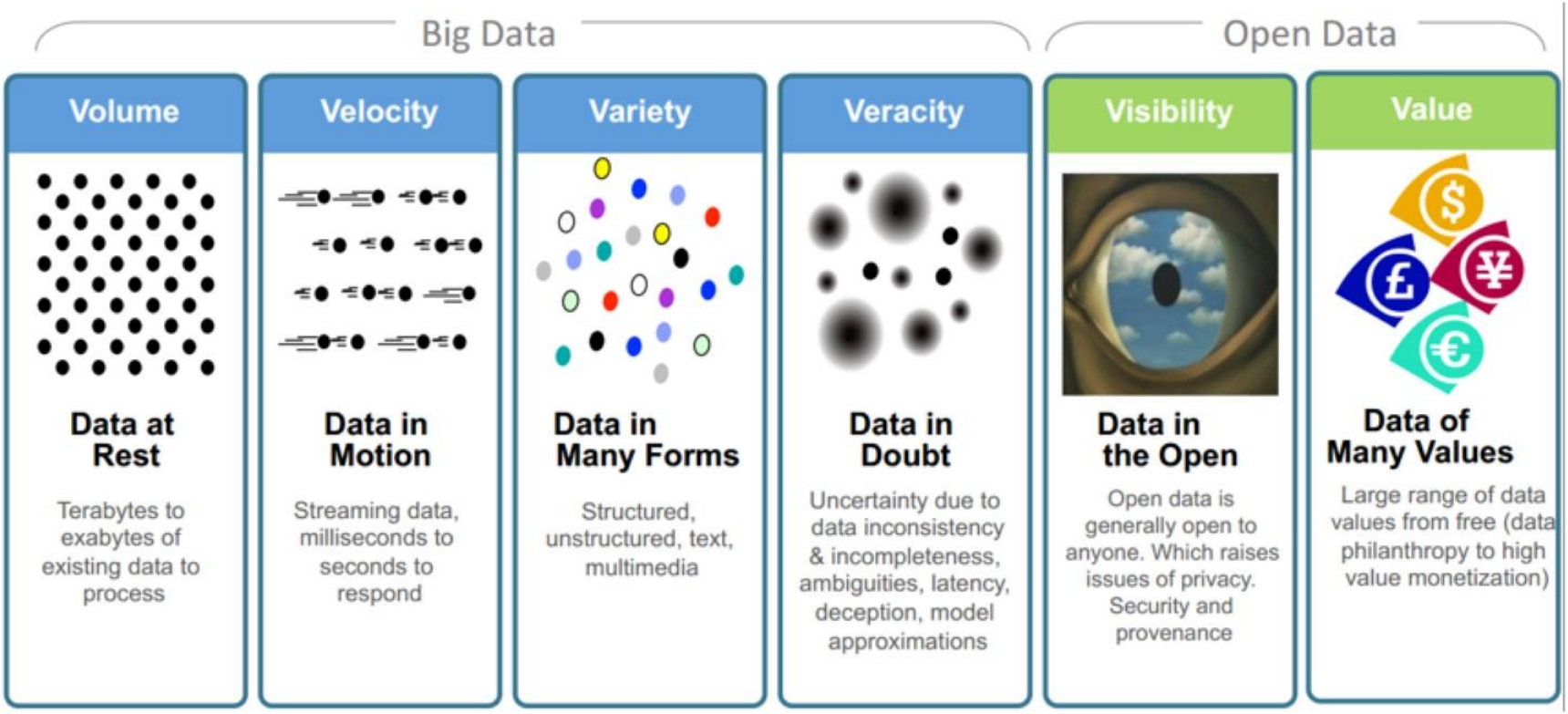
Introduction to Basics of Big Data

Big Data: Vs??????????

Big Data: 6V in Summary



Transforming Energy and Utilities through Big Data & Analytics. By Anders Quitzau@IBM

Other V's

- **Variability**
  - Variability refers to data whose meaning is constantly changing. This is particularly the case when gathering data relies on language processing.
- **Viscosity**
  - This term is sometimes used to describe the latency or lag time in the data relative to the event being described. We found that this is just as easily understood as an element of Velocity.
- **Virality**
  - Defined by some users as the rate at which the data spreads; how often it is picked up and repeated by other users or events.
- **Volatility**
  - Big data volatility refers to how long is data valid and how long should it be stored. You need to determine at what point is data no longer relevant to the current analysis.
- More V's in the future …

Types of Big Data

Structured Data

Examples Of Structured Data

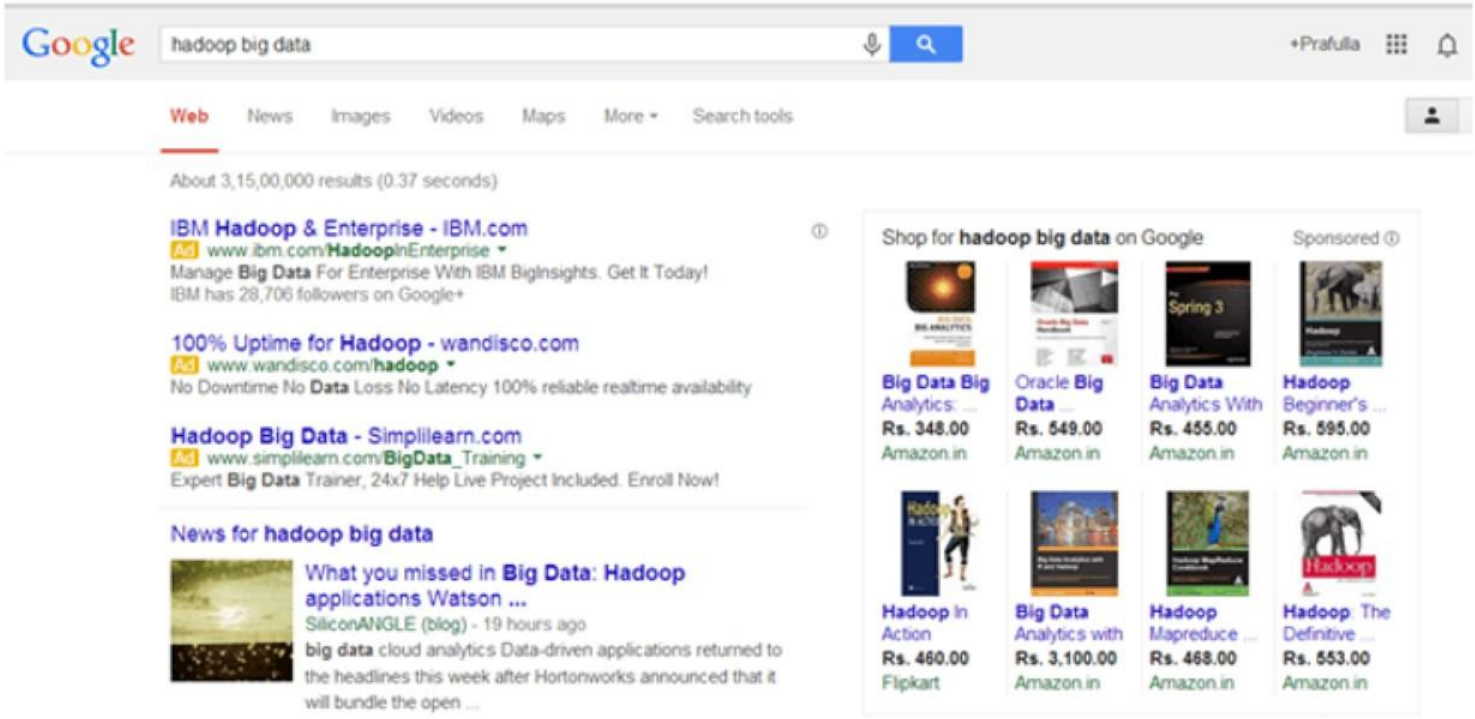
An 'Employee' table in a database is an example of Structured Data

Employee_ID	Employee_Name	Gender	Department	Salary_In_lacs
2365	Rajesh Kulkarni	Male	Finance	650000
3398	Pratibha Joshi	Female	Admin	650000
7465	Shushil Roy	Male	Admin	500000
7500	Shubhojit Das	Male	Finance	500000
7699	Priya Sane	Female	Finance	550000

Types of Big Data

Un-structured Data

- The output returned by 'Google Search'



Types of Big Data

Semi-structured Data

- Personal data stored in an XML file-

```
<rec><name>Prashant Rao</name><sex>Male</sex><age>35</age></rec>
<rec><name>Seema R.</name><sex>Female</sex><age>41</age></rec>
<rec><name>Satish Mane</name><sex>Male</sex><age>29</age></rec>
<rec><name>Subrato Roy</name><sex>Male</sex><age>26</age></rec>
<rec><name>Jeremiah J.</name><sex>Male</sex><age>35</age></rec>
```

Can you spot some difference between Traditional Analytics and Big Data Analytics ?

Big Data Formats

Consideration 1: ROW VS. COLUMN

- It is the most important consideration.
- A row based format or a column-based format.
- At the highest level, column-based storage is most useful when performing analytics queries that require only a subset of columns examined over very large data sets.
- If your queries require access to all or most of the columns of each row of data, row-based storage will be better suited to your needs.

Big Data Formats

- To help illustrate the differences between row and column-based data, consider this table of basic transaction data.
- For each transaction, we have the customer name, the product ID, sale amount, and the date.

EXAMPLE: SAMPLE TRANSACTION DATA

Customer Name	Product ID	Sale Amount	Transaction Date
Emma	Prod 1	100.00	2018-04-02
Liam	Prod 2	79.99	2018-04-02
Noah	Prod 3	19.99	2018-04-01
Olivia	Prod 2	79.99	2018-04-03

Big Data Formats

- **Row-based storage** is the simplest form of data table and is used in many applications, from web log files to highly-structured database systems like MySql and Oracle.

EXAMPLE: SAMPLE TRANSACTION DATA

Customer Name	Product ID	Sale Amount	Transaction Date
Emma	Prod 1	100.00	2018-04-02
Liam	Prod 2	79.99	2018-04-02
Noah	Prod 3	19.99	2018-04-01
Olivia	Prod 2	79.99	2018-04-03

```
Emma,Prod1,100.00,2018-04-02;Liam,Prod2,79.99,2018-04-02;Noah,Prod3,19.99,2018-04-01;Olivia,Prod2,79.99,2018-04-03
```

Big Data Formats

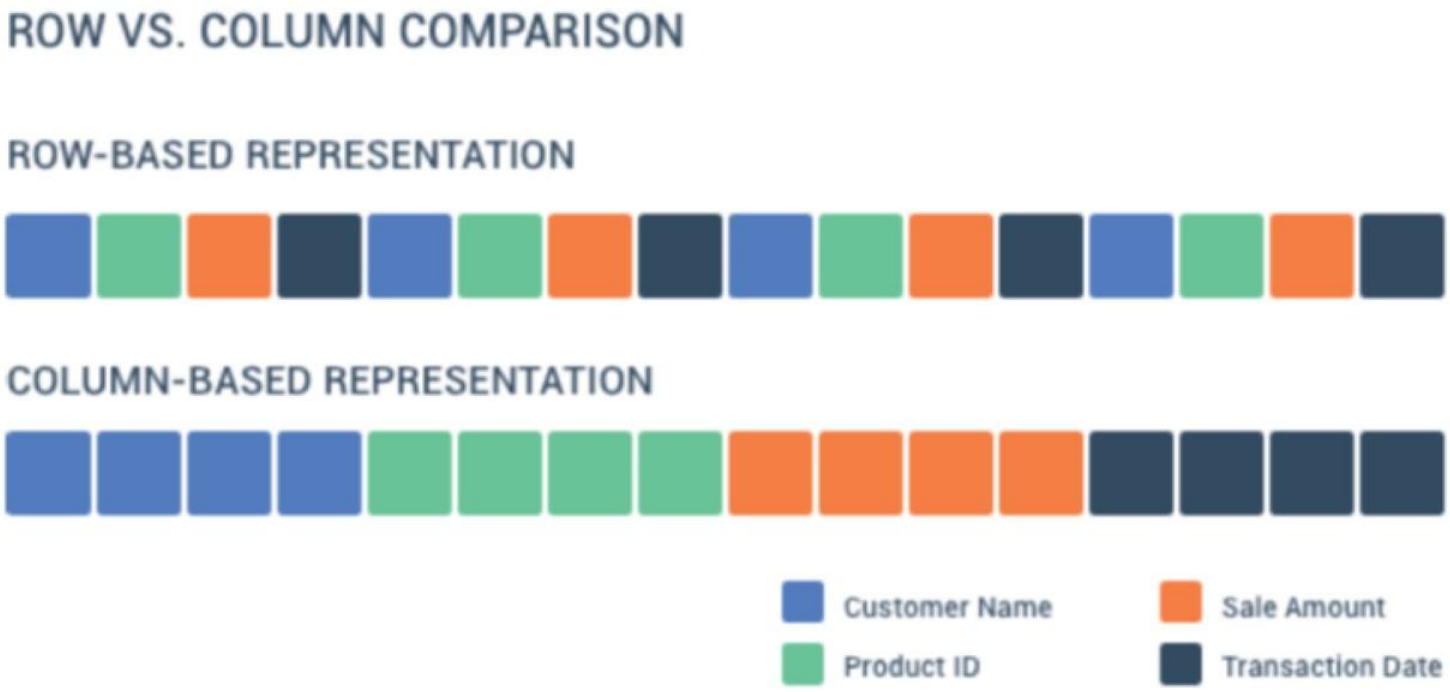
- In columnar formats, data is stored sequentially by column, from top to bottom—not by row, left to right. Having data grouped by column makes it more efficient to easily focus computation on specific columns of data. Column-based storage is also ideal for sparse data sets where you may have empty values.

EXAMPLE: SAMPLE TRANSACTION DATA

Customer Name	Product ID	Sale Amount	Transaction Date
Emma	Prod 1	100.00	2018-04-02
Liam	Prod 2	79.99	2018-04-02
Noah	Prod 3	19.99	2018-04-01
Olivia	Prod 2	79.99	2018-04-03

```
Emma,Liam,Noah,Olivia;Prod1,Prod2,Prod3;Pr
od2;100.00,79.99,19.99,79.99;2018-04-02,2018-04-02,
2018-04-01, 2018-04-03
```

Big Data Formats



To analyze “Sale Amount” (orange) and “Transaction Date” (navy blue) Column Based format is better than Row Based format

Big Data Formats

Consideration 2: **SCHEMA EVOLUTION**

- "Schema" in a database context, is its organization—the tables, columns, views, primary keys, relationships, etc.
- When evaluating schema, there are a few key questions to ask of any data format:
  - How easy is it to update a schema (such as adding a field, removing or renaming a field)?
  - How will different versions of the schema "talk" to each other?
  - Is it human-readable? Does it need to be?
  - How fast can the schema be processed?
  - How does it impact the size of data?

Big Data Formats

Consideration 3: **SPLITABILITY**

- Processing such datasets efficiently usually requires breaking the job up into parts that can be farmed out to separate processors.
- In fact, large-scale parallelization of processing is key to performance.
- Your choice of file format can critically affect the ease with which this parallelization can be implemented.
- For example, if each file in your dataset contains one massive XML structure or JSON record, the files will not be “splittable”, i.e. decomposable into smaller records that can be handled independently.

Big Data Formats

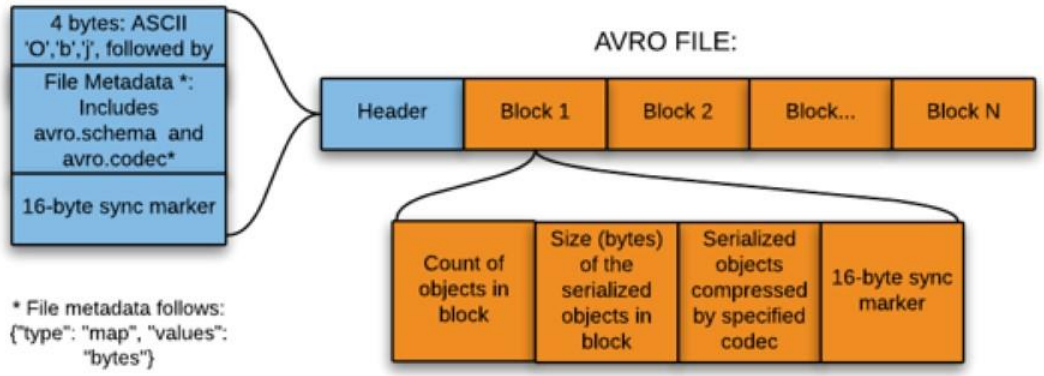
Consideration 4: **COMPRESSION**

- Data compression reduces the amount of information, the resources required to store and transmit data, typically saving time and money.
- Columnar data can achieve better compression rates than row-based data.
- Storing values by column, with the same type next to each other, allows you to do more efficient compression on them than if you're storing rows of data.
- For example, storing all dates together in memory allows for more efficient compression than storing data of various types next to each other—such as string, number, date, string, date.

Big Data Formats

APACHE AVRO: **A ROW BASED FORMAT**

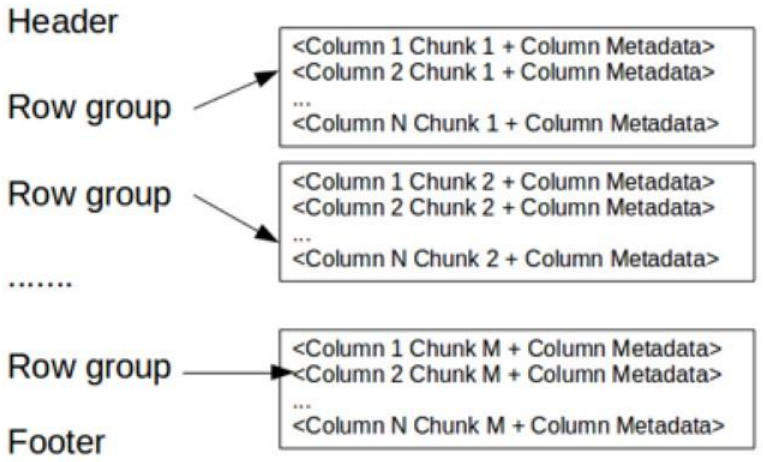
- Released by the Hadoop working group in 2009, is highly splittable.
- Schema travels with data.
- The data definition is stored in JSON format while the data is stored in binary format, minimizing file size and maximizing efficiency.
- Easy to change.



Big Data Formats

APACHE PARQUET: **A COLUMN BASED FORMAT**

- Launched in 2013, Parquet was developed by Cloudera and Twitter to serve as an optimized columnar data store on Hadoop.
- Because data is stored by columns, it can be highly compressed and splittable (for the reasons noted above).
- The column metadata for a Parquet file is stored at the end of the file, which allows for fast, one-pass writing. Metadata can include information such as, data types, compression/encoding scheme used (if any), statistics, element names, and more.



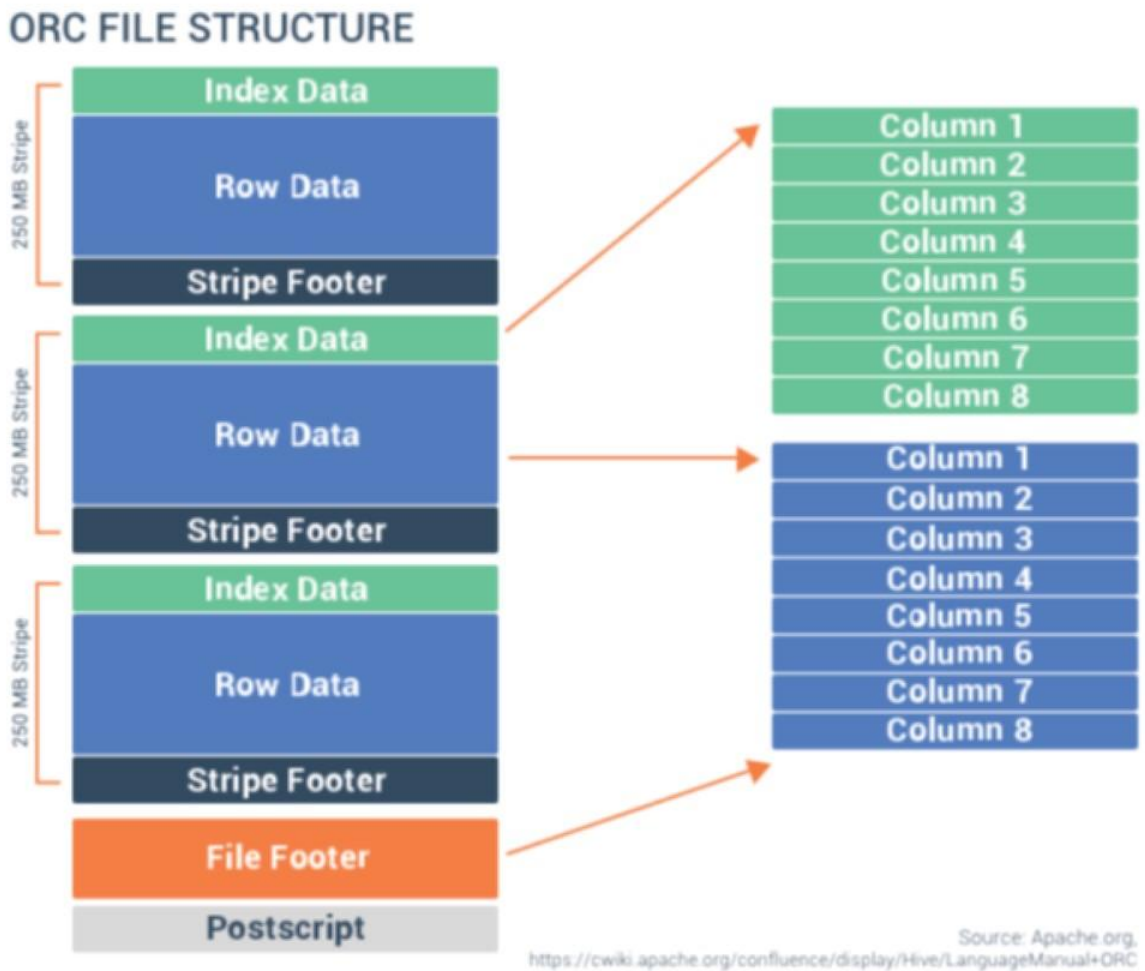
Big Data Formats

APACHE ORC: **A ROW-COLUMNAR BASED FORMAT**

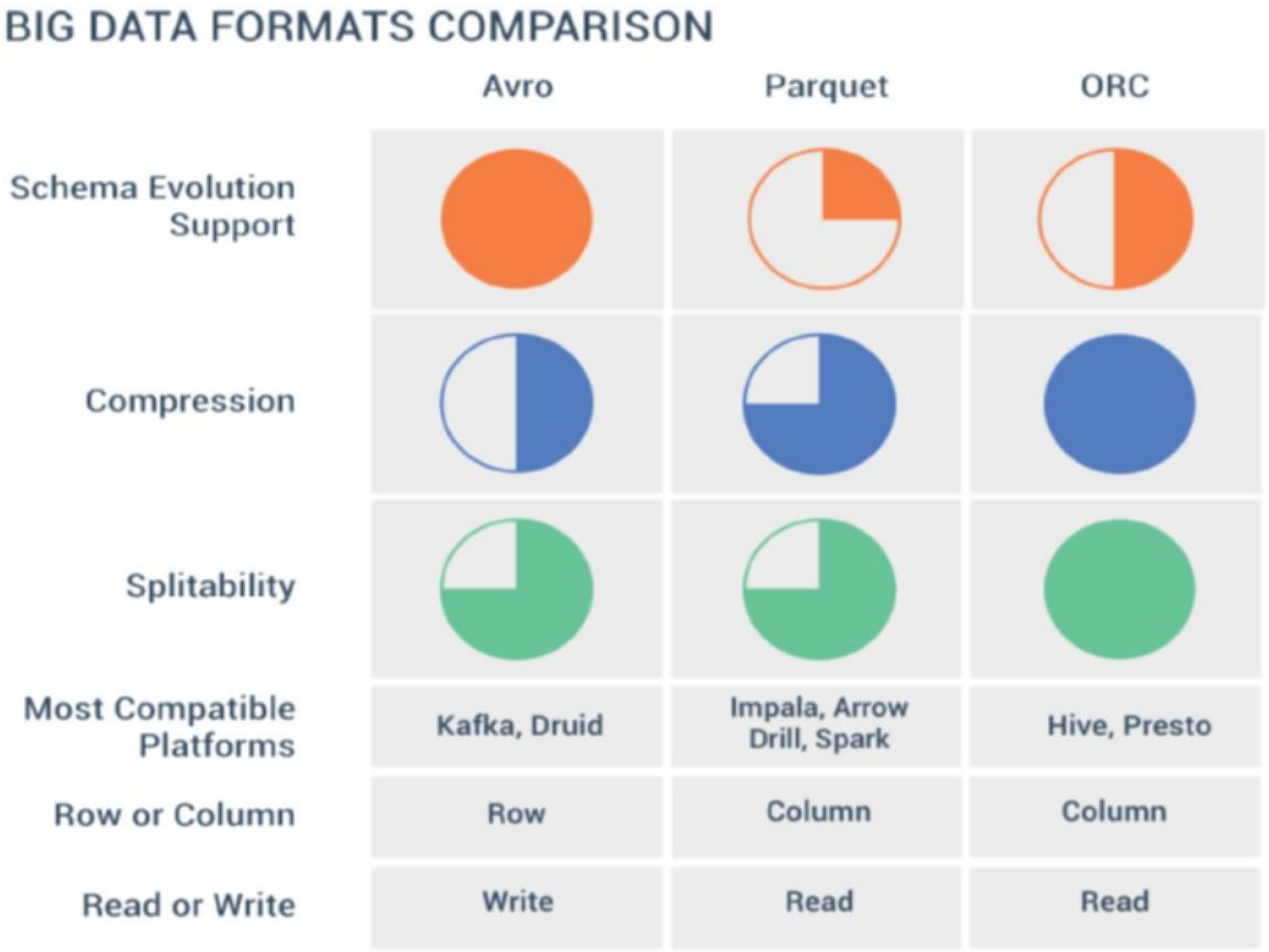
- Optimized Row Columnar (ORC) format was first developed at Hortonworks to optimize storage and performance in Hive, a data warehouse for summarization, query and analysis that lives on top of Hadoop.
- Hive is designed for queries and analysis, and uses the query language HiveQL (similar to SQL). ORC files are designed for high performance when Hive is reading, writing, and processing data.
- ORC stores row data in columnar format. This row-columnar format is highly efficient for compression and storage.
- It allows for parallel processing across a cluster, and the columnar format allows for skipping of unneeded columns for faster processing and decompression.

Big Data Formats

- APACHE ORC: **A ROW-COLUMNAR BASED FORMAT**



Big Data Format Comparison



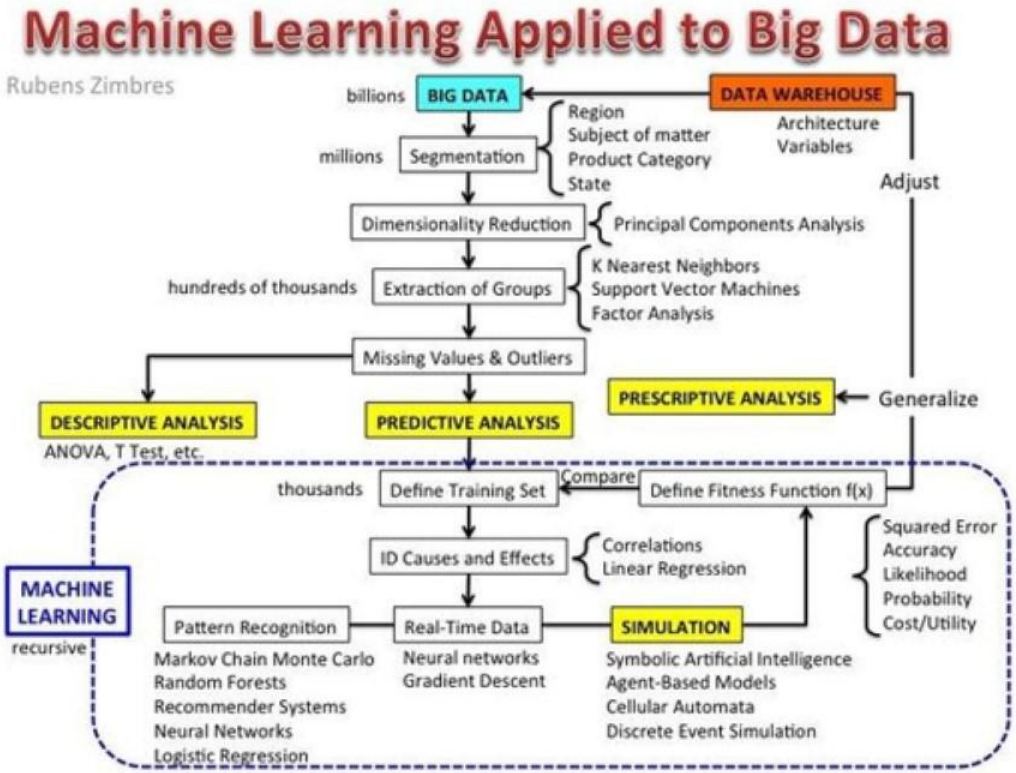
Big Data and Machine Learning

Why Big Data need different types of Machine Learning algorithms

- Learning for different types of data
- Learning for high speed of streaming data
- Learning for uncertain and incomplete data
- Learning for data with low value density and meaning diversity

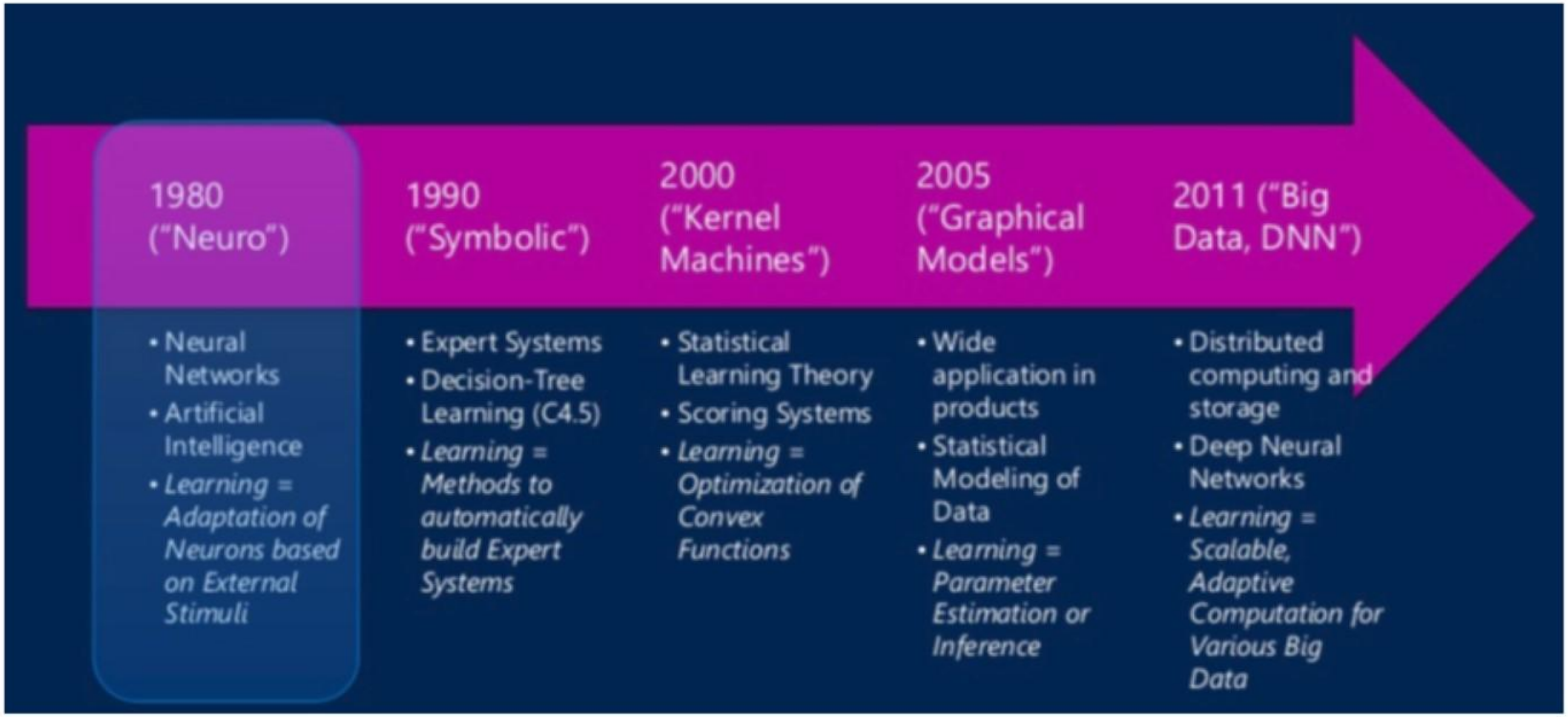
Machine Learning for Big Data

- Rubens Zimbres captures the synergistic interaction between machine learning and big data and is shown as a mindmap as follows:

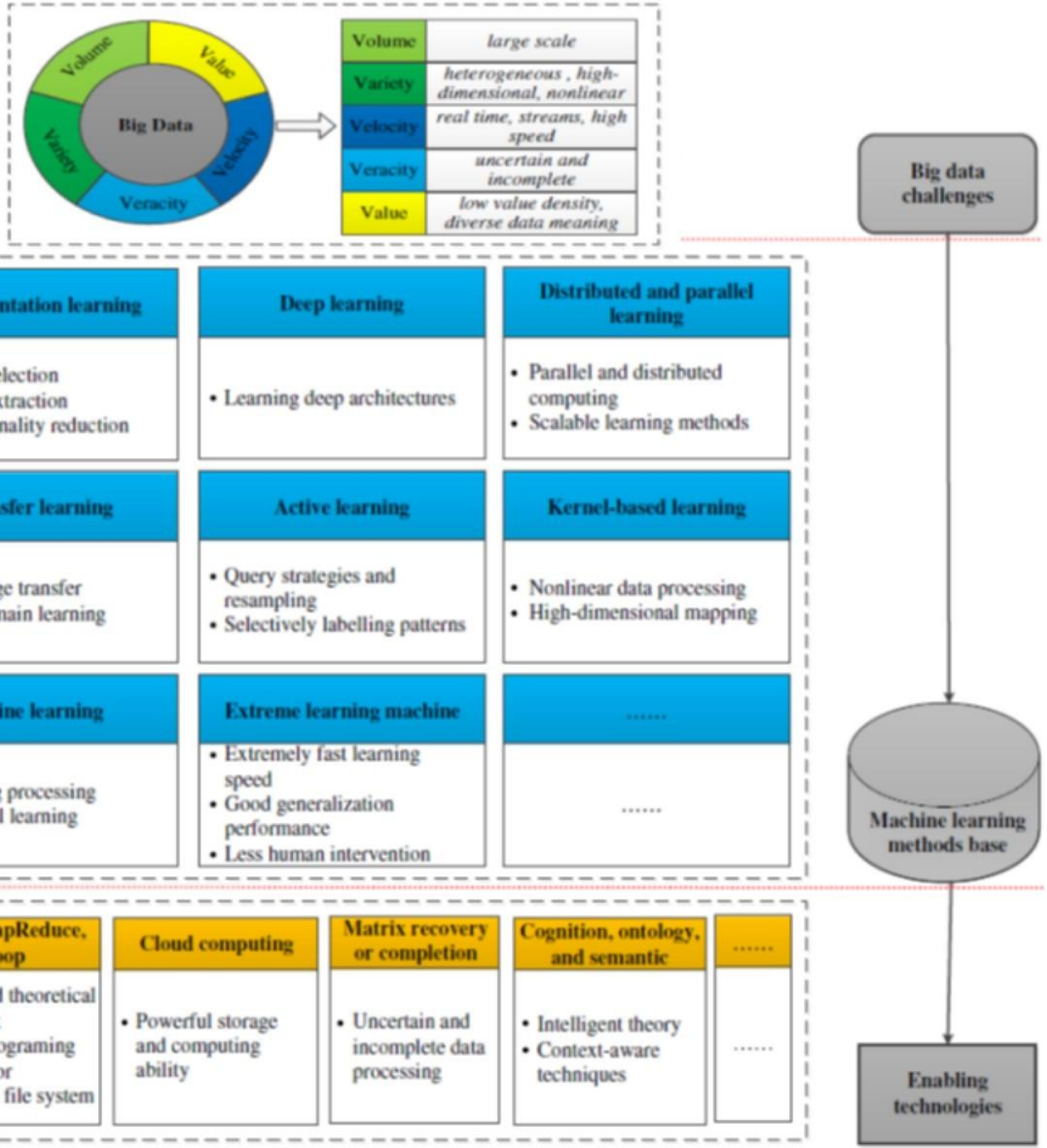


https://www.datasciencecentral.com/profiles/blogs/key-tools-of-big-data-for-transformation-review-case-study

Machine Learning Advancement and Big Data



Hierarchical framework of efficient machine learning for big data processing



Big Data Challenges and Machine Learning Advantages

Big Data Challenges: Volume

- According to a recent study by (MIT), 1 the average processing speed in humans is around 45 bits per second, with skilled individuals going up to 60 bits per second.
- That means that if a person were to spend 24 hours a day just reading (without ever stopping, not even to eat or drink or sleep), that person would process slightly above 5 MB of data.
- It would take such a person around 536 days to go through a 1TB data set – and at the end of the process, the person would very likely not remember the entire data set.

Machine Learning Advantages

- ML has no issue with volume; in fact, the more data you have, the higher-quality results you expect.
- In terms of capability to process volume, the machine is constrained only by the size of the machine (a modern laptop can likely read 1TB data in a few seconds)

Big Data Challenges and Machine Learning Advantages

Big Data Challenges: Velocity

- Data changes too quickly for a human to understand the consequences and react in a positive manner.
- For example, the New York Stock Exchange typically produces 1 TB of trade information during each trading session.
- In simplistic terms, 1 TB is equal to a couple million decent-sized books. Humans simply cannot absorb the data fast enough and, even more important, are unable to sift through the data to find thevaluable actionable insight

Machine Learning Advantages

- ML operates at machine speeds: a successfully trained model can react to input data at speeds impossible for a human.
- Recent advances in neural nets allow for immediate reaction to inputs "close to edge"– where the event happens.
- Reaction times of an efficient model can be measured in tens of milliseconds.

Big Data Challenges and Machine Learning Advantages

Big Data Challenges :Variety

- Most enterprises now have data from multiple sources (e.g., databases, real-time social media streams, IoT reporting) and in various formats (e.g., structured, unstructured).
- The average human intellect will almost certainly be unable to construct and hold in memory a data model the considers every single possible correlation and relationship among data from multiple sources and in multiple formats.
- The variety of data needs to be simplified for human consumption either via description, aggregation, or summarization, but simplification loses the potential of big data and makes finding hidden insights unlikely

Machine Learning Advantages

- Variety is not a problem for machines, as extremely complex models can be built and held in machine memory.
- ML operates at machine scale with modern machines capable of holding and making use of terabytes of information in milliseconds

Big Data Challenges and Machine Learning Advantages

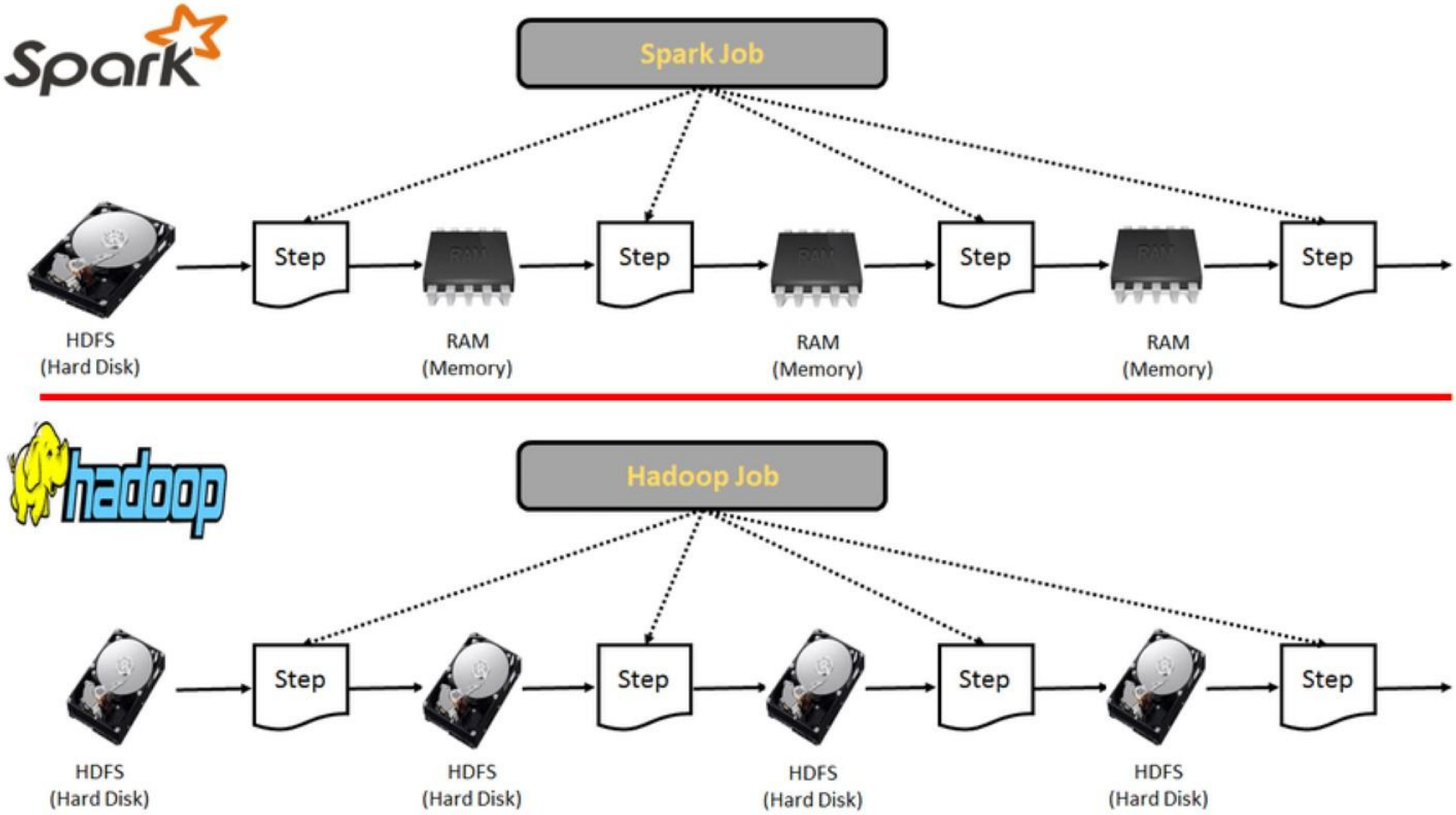
Big Data Challenges :Veracity

- We cannot be confident that the data we have is clean, usable, and of high quality – and even worse, humans are prone to errors of judgement such as confirmation bias and will tend to attribute more worth to data that confirms biases – even if it means losing out financially

Machine Learning Advantages

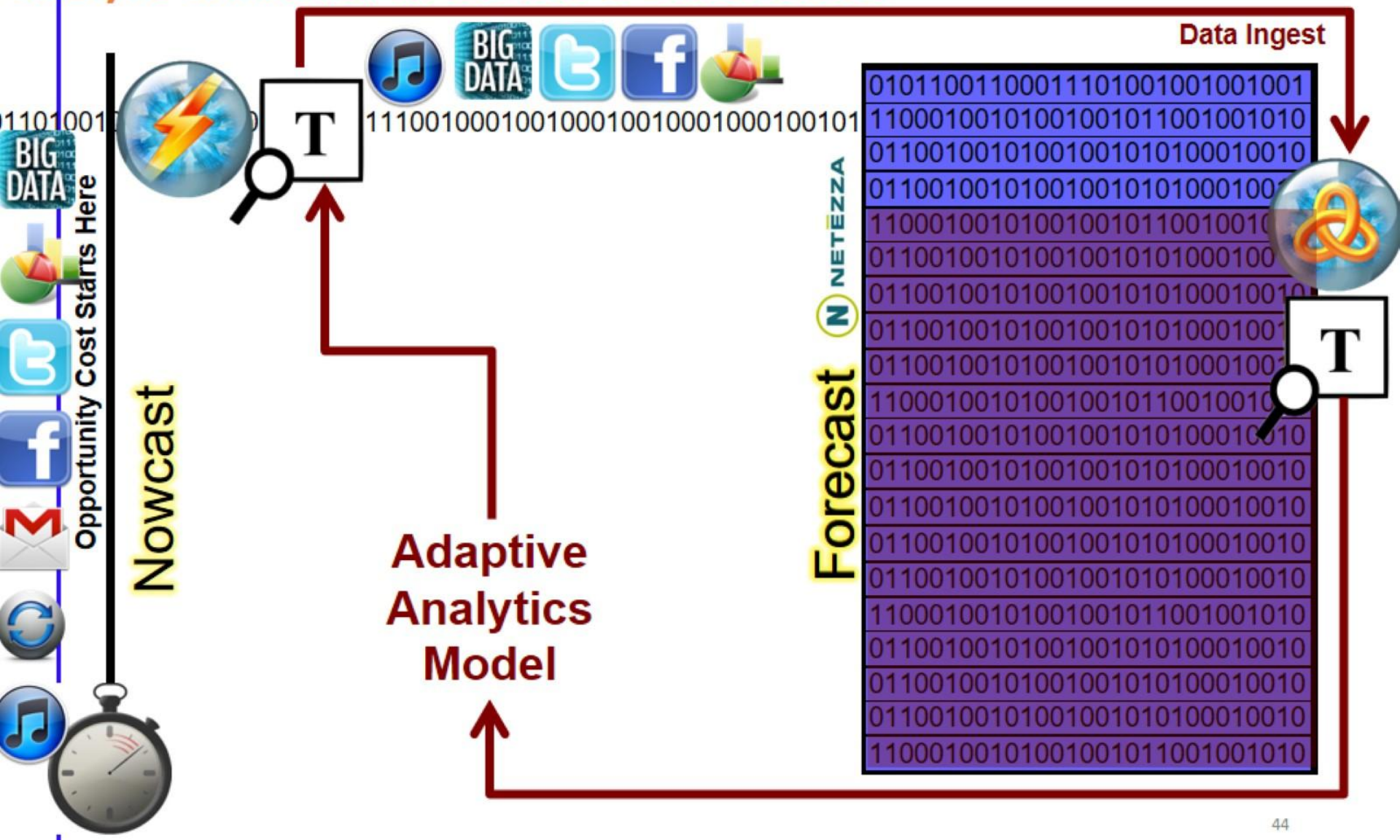
- ML has no confirmation (or other) bias when analysing data.
- Proper algorithms and models will also trend to recognize "bad" data.
- Significant advances in automated algorithms exist that allow the machine to effectively ignore noise and outliers.

In Memory Versus Disk based Computation



BIG DATA STREAM ANALYTICS

Analytic With *Data-In-Motion & Data At Rest*



Background

“The only knowledge that can hurt you is the knowledge you don't have.”

-unknown

Knowledge you don’t have = Variation

Background

- Variation is present in all processes.
- Variation is additive--variation in the process inputs will generate more variation in the process output.

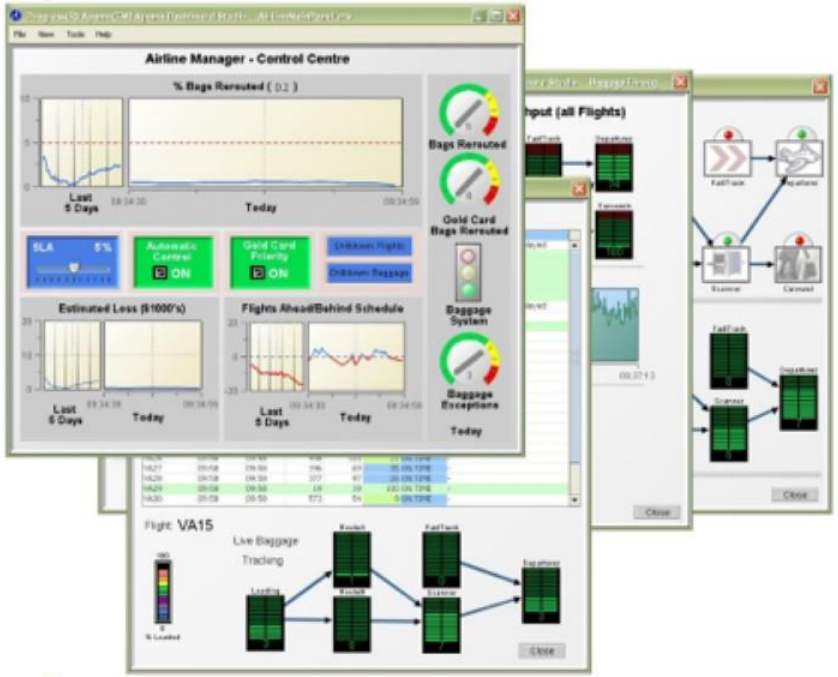
Variation exists everywhere!

Goal is to Reduce the variation and to make sources of variation faster to remedy ….As variation decreases, profitability increases

Organizations Wish to Make Decisions Faster

Real-Time Compliance

Monitor and stop out-of-compliance activity *before it happens*, impacts operations, incurs penalties



Travel

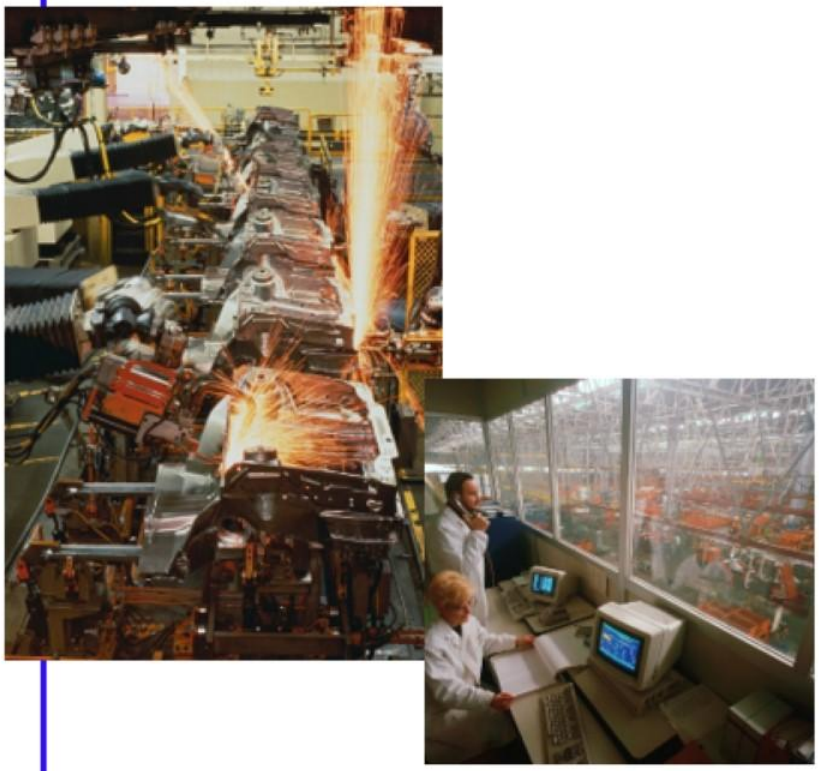
Monitor airport operations, baggage movement, weather reports to ensure all flights are transitioned in and out efficiently

Organizations Wish to Make Decisions Faster

Supply Chain & Retail Optimization

Automated item level inventory tracking from supply chain to retail operations.

Where is this book now?



Manufacturing

Are all lines operating at maximum efficiency?

Is a downstream problem in one line going to have an adverse affect in upstream processes?

Can I absorb last minute changes in demand?

Example: Truck Logistics

CEP: The Brains of BAM

"**When** any truck has consumable cargo that is **> 1 hour** behind schedule and **temp > 30** degrees, **alert** the operator with a priority 1 exception."

Application

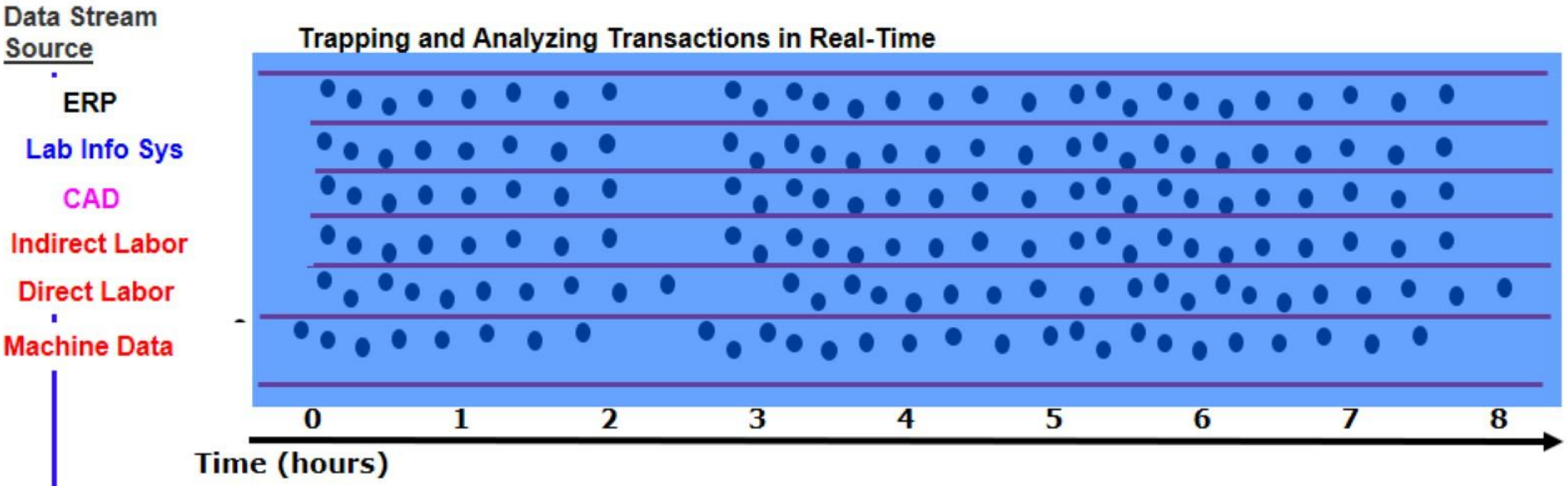
- GPS and cellular networks identify truck locations with exceptions reported to management if journey times differ from historic norms.
- CEP rules defined to detect deviations from schedule

Benefits

- Better utilisation of resources
- Insight can be given to customers on expected arrival times
- Automate intelligent predictive behaviour

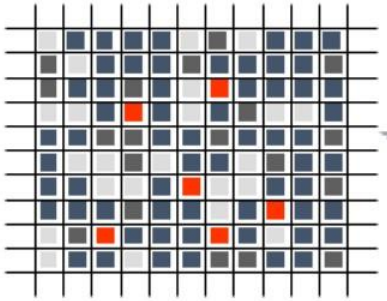
– Source: www.progress.com/apama

Real Time Data Stream

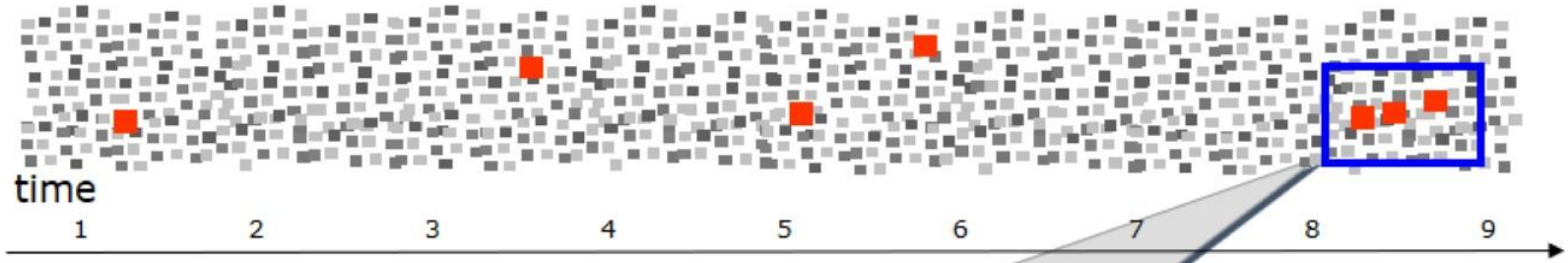


Complex Event Processing

Flipping traditional computing on its head

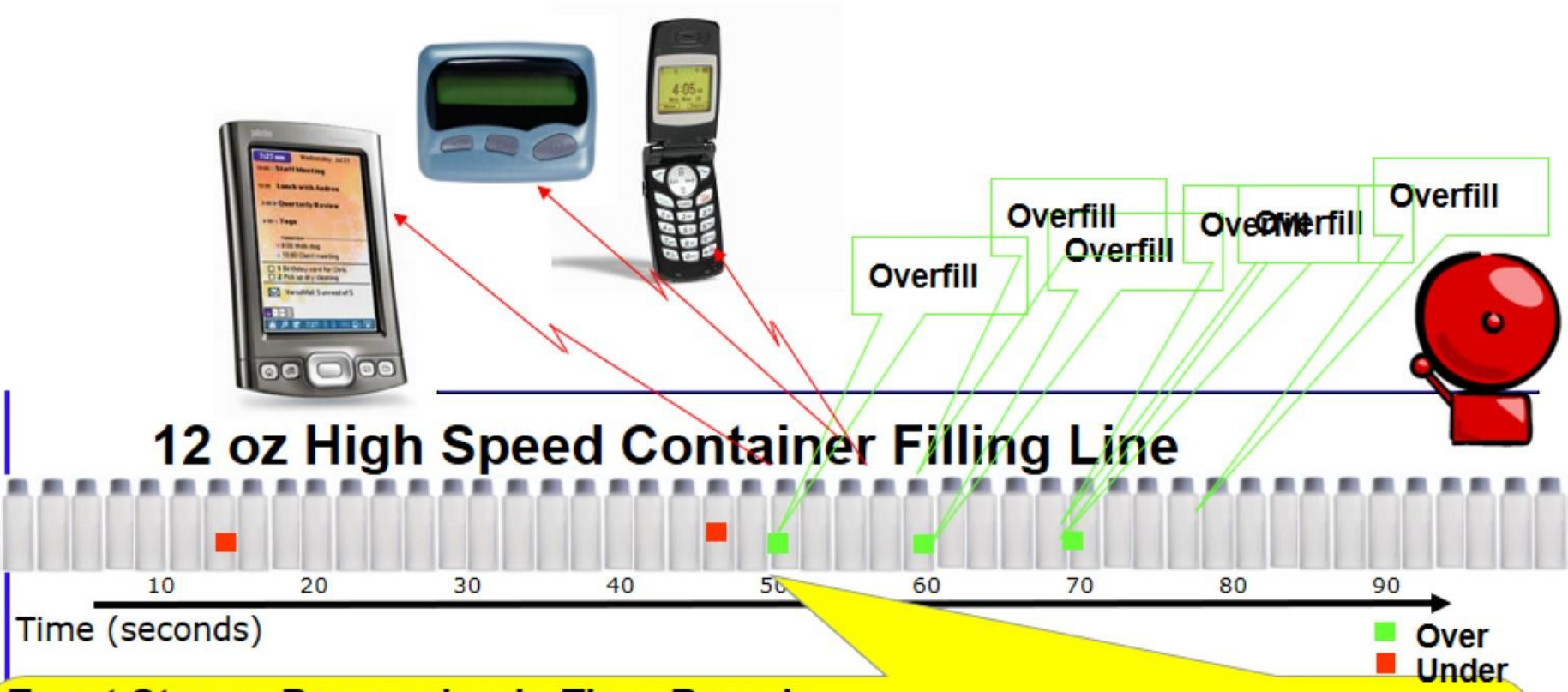


Static Data Processing: "How many out containers did we fill yesterday? This week? Last month?



Complex Event Processing: "When 3 tubes report 10% or greater deviance from expected fill within any 30 second window, log alert. If any 10 tubes subsequently also out of range in the next 60 second, dispatch maintenance."

Complex Event Processing



Event Stream Processing is Time-Based:

Example:

"If any 2 consecutive container's fill weight varies from its standard by <.02 oz (Underfilled) OR any 3 containers >.01 oz (Overfilled) in any 30-second window, sound an alarm, and page a process engineer."

Application Event Monitoring

Machine out of spec *...5 minutes* *...request maintenance*

Order 1111 priority raised *...10 minutes* *...adjust schedule?*

Maintenance report *...10 minutes* *...setup new line*

Complex Event Processing

Messaging Infrastructure

Alerts

ERP

Plant Floor

Maintenance

Monitoring Existing Applications in Real Time

Why Data Stream Processing is important

- Real-time visibility of any system help us to become more proactive
- To avoid delayed and inaccurate data
- To avoid machine down time
- To achieve machine utilization and efficiency
- To have better asset preventative maintenance

Understanding practical constraints using simple streaming example

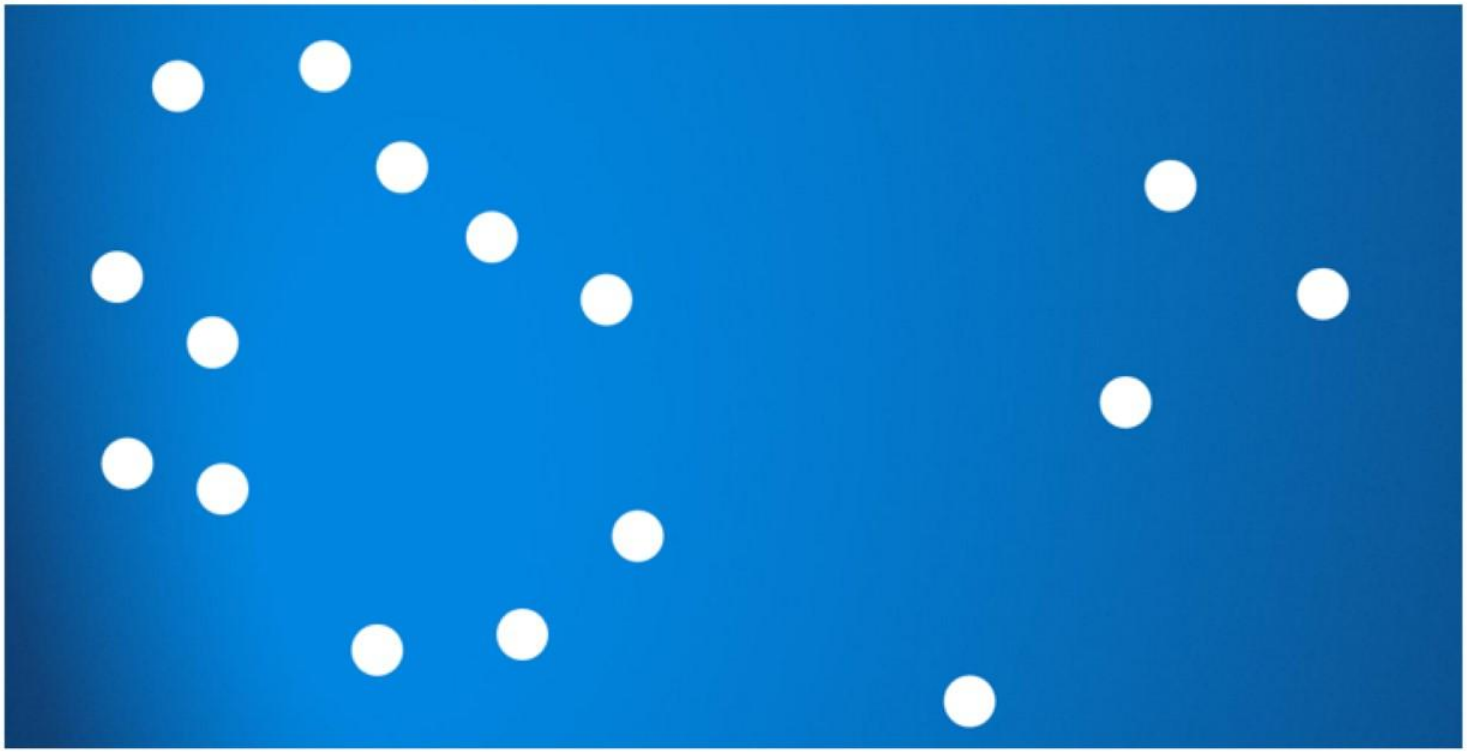
- How many dots can you see?
- (Each dot is an event)

Understanding practical constraints using simple streaming example

- How many dots can you see?
- (Each dot is an event)

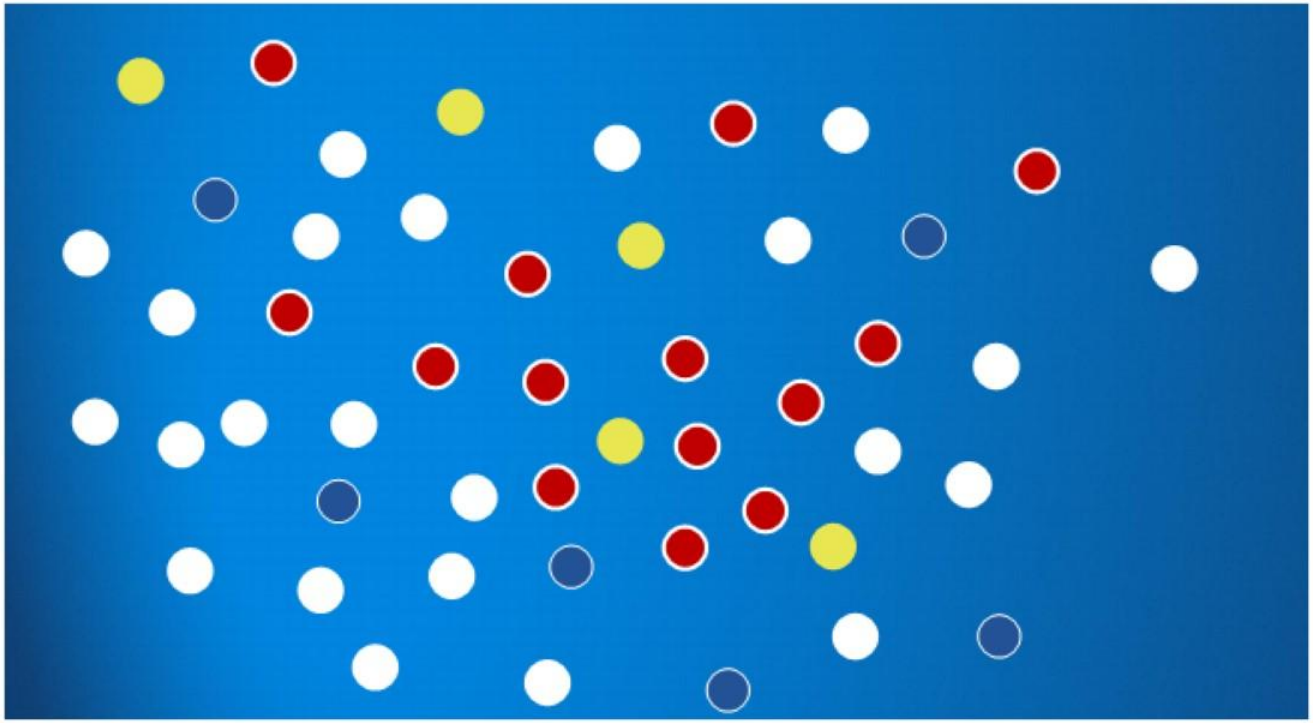
Understanding practical constraints using simple streaming example

- How many dots can you see?
- (Each dot is an event)



Let’s add some complexity

- How many red dots can you see?



Let's add some temporal complexity

- How many yellow dots appear after the blue dots?
- How many red dots will there be next?
- IF there were 20 red dots, put the next three blue dots in a specific category.
- If there are ten blue dots, join to alternate data and see how many blue dots there are in total ?
- If there were 6 green dots, wait for an hour, then compare again.

Data Stream

Definition

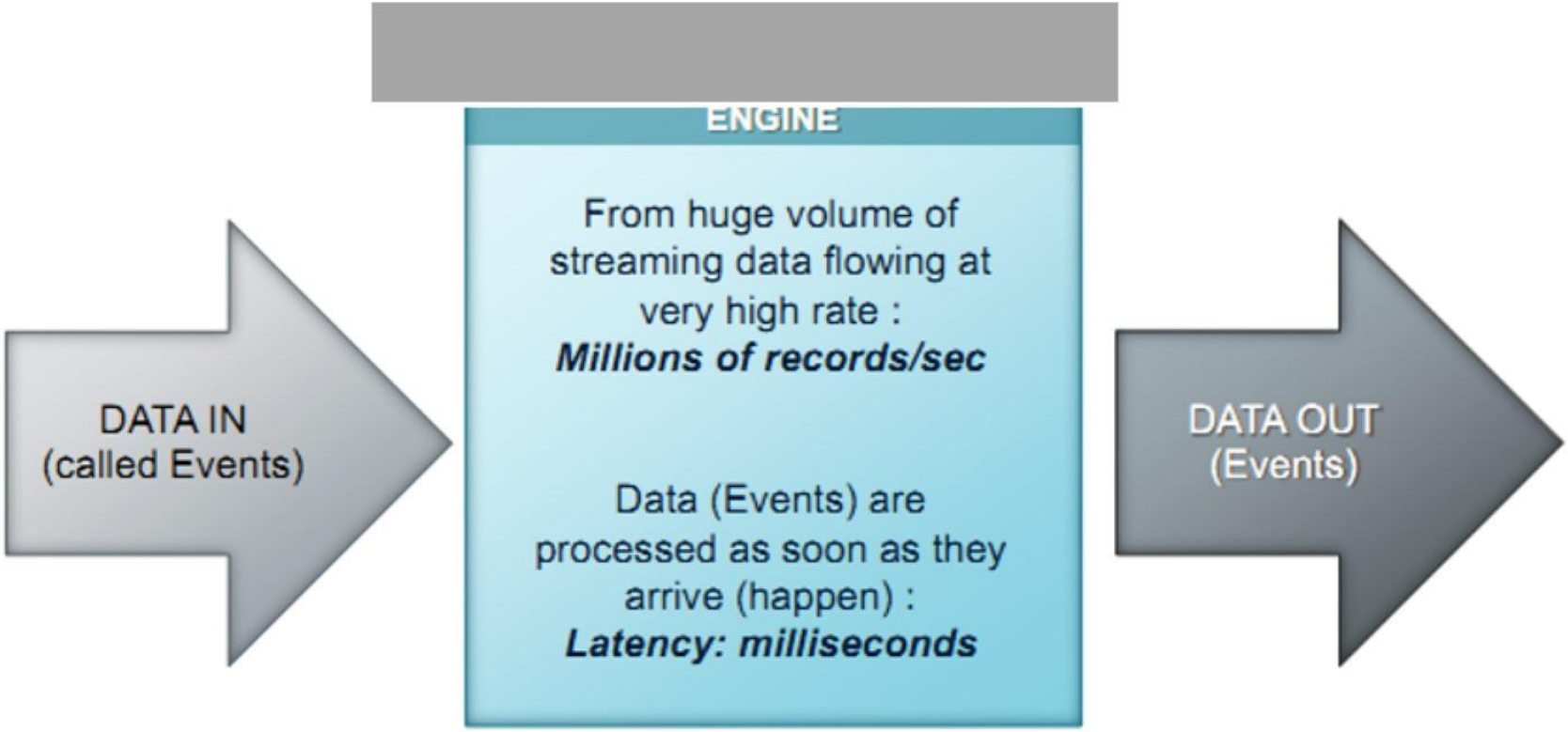
A data stream is an ordered and potentially infinite sequence of data points:

$$\langle \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \ldots \rangle$$

where $\mathbf{y}_i$ is a tuple (e.g., a vector)

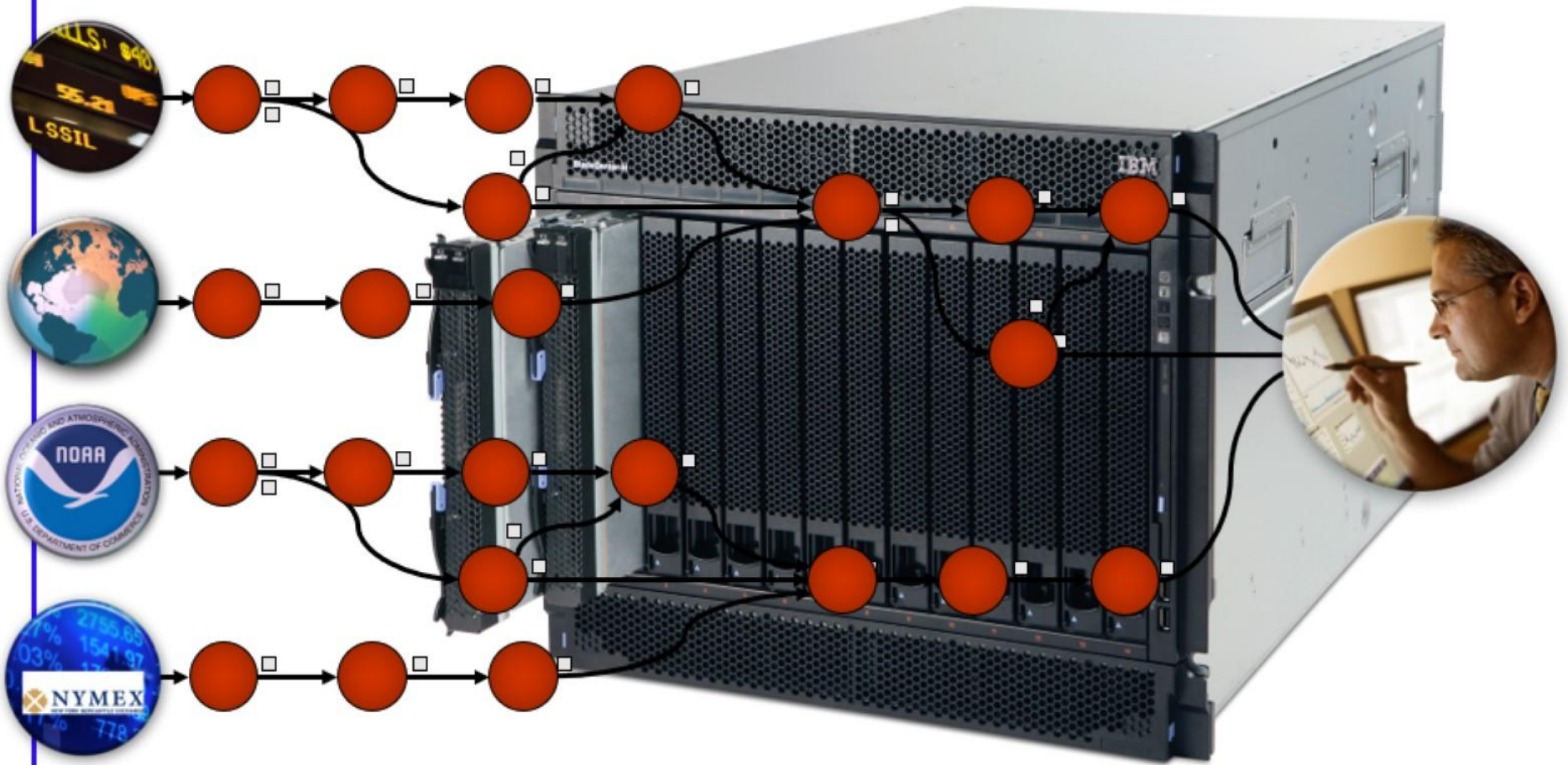
- real time industrial control system dataset

Handling of Stream of Events



What is Stream Computing?

Continuous Ingestion → Continuous Analysis in Microseconds

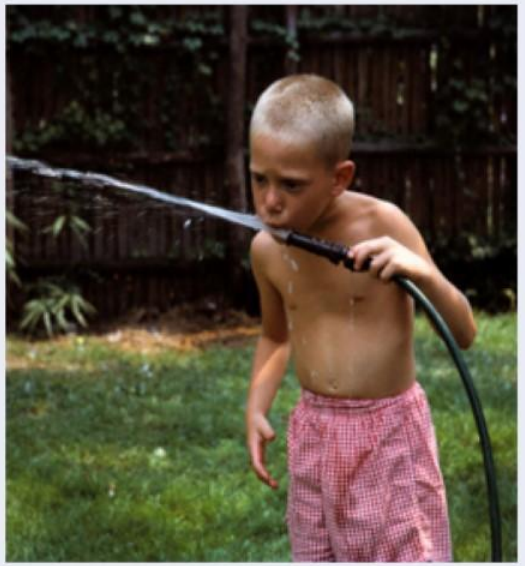


Characteristics of Data Streams

- Data Streams
  - **Data streams**—continuous, ordered, changing, fast, huge amount
  - **Traditional DBMS**—data stored in finite, persistent data sets
- Characteristics
  - Huge volumes of continuous data, possibly infinite
  - Fast changing and requires fast, real-time response
  - Data stream captures nicely our data processing needs of today
  - Random access is expensive—single scan algorithm (*can only have one look*)
  - Store only the summary of the data seen thus far
  - **Most stream data are at pretty low-level or multi-dimensional in nature, needs multi-level and multi-dimensional processing**

Batch Processing Vs Stream Processing

Batch Processing	Stream Processing
High-latency apps	Low-latency apps
Static Files	Event Streams
Process-after-store	Sense-and-respond
Batch processors	Stream processors



Batch Processing Vs Stream Processing

Batch Processing

- Persistent relations
- One-time queries
- Random access
- “Unbounded” disk store
- Only current state matters
- No real-time services
- Relatively low update rate
- Data at any granularity
- Assume precise data
- Access plan determined by query processor, physical DB design

Stream Processing

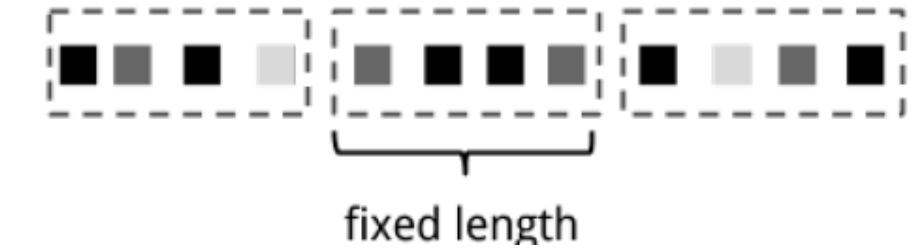
- Transient streams
- Continuous queries
- Sequential access
- Bounded main memory
- Historical data is important
- Real-time requirements
- Possibly multi-GB arrival rate
- Data at fine granularity
- Data stale/imprecise
- Unpredictable/variable data arrival and characteristics

Windows in Stream Analytics

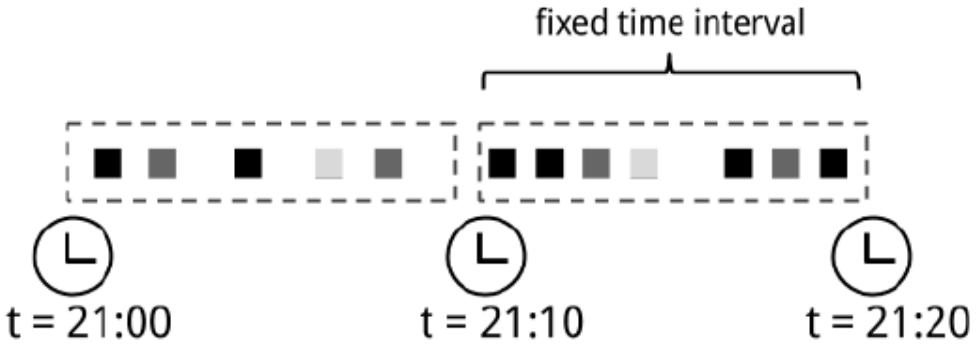
- Mechanism for extracting a finite relation from an infinite stream
- Various window proposals for restricting operator scope.
  - Windows based on ordering attribute (e.g. time)
  - Windows based on tuple counts
  - Windows based on explicit markers (e.g. punctuations)
  - Variants (e.g., partitioning tuples in a window)

Types of Windows

- **Tumbling** windows assign events into non-overlapping buckets of fixed size.



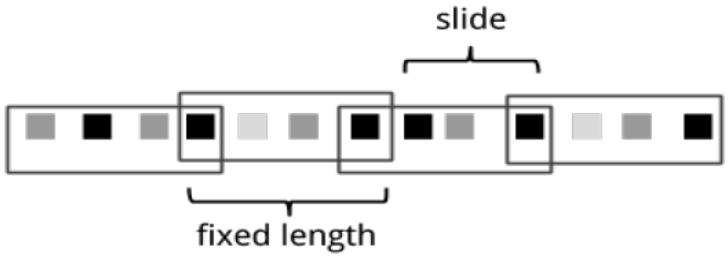
Count Based Window



Time Based Window

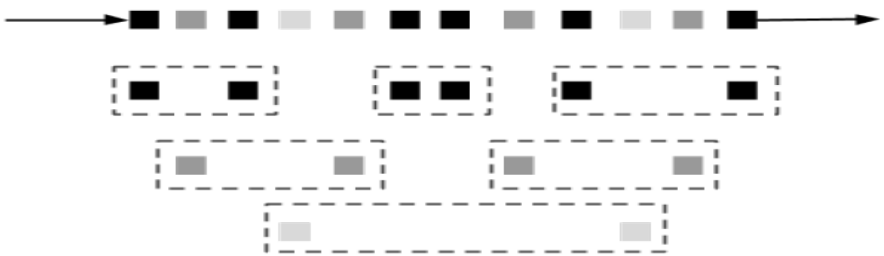
Types of Windows

- **Sliding** windows assign events into overlapping buckets of fixed size.



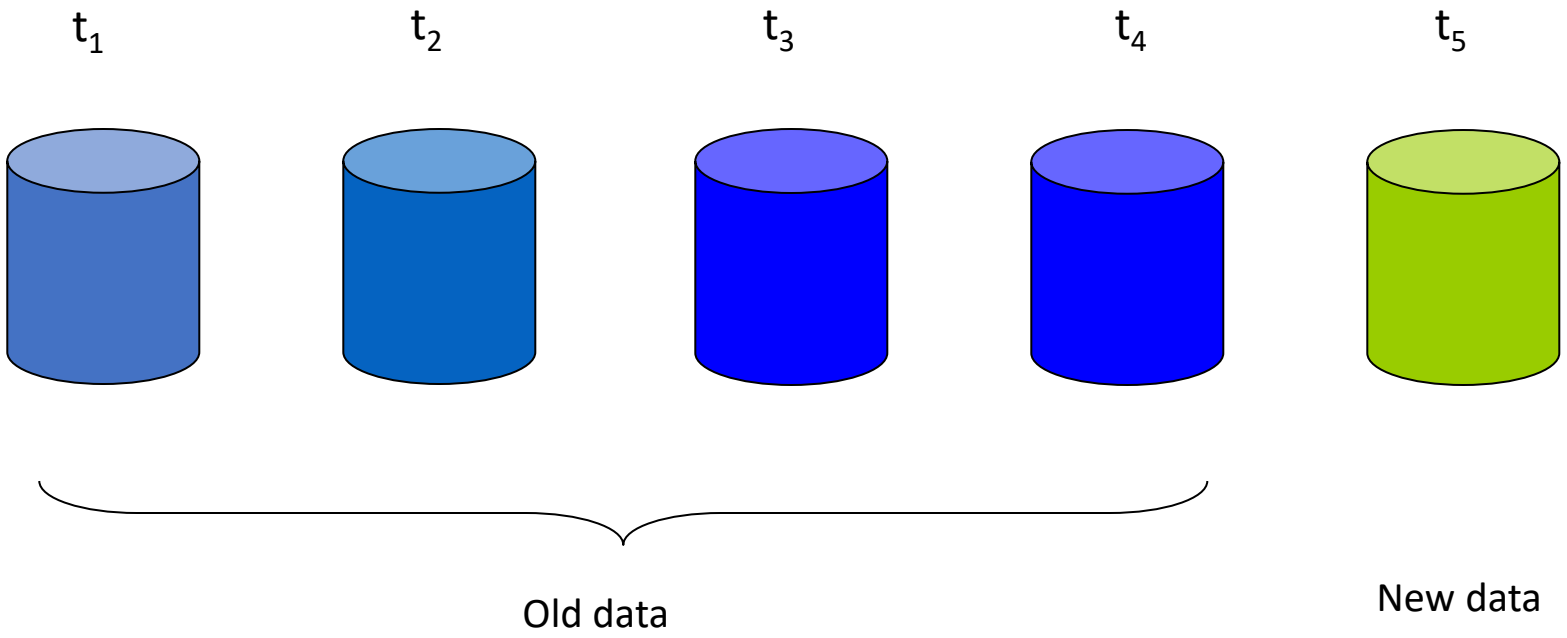
Sliding Window

Session Window



A parallel count-based tumbling window of length 2.

Challenges in Data Streams



Characteristics: may change over time.

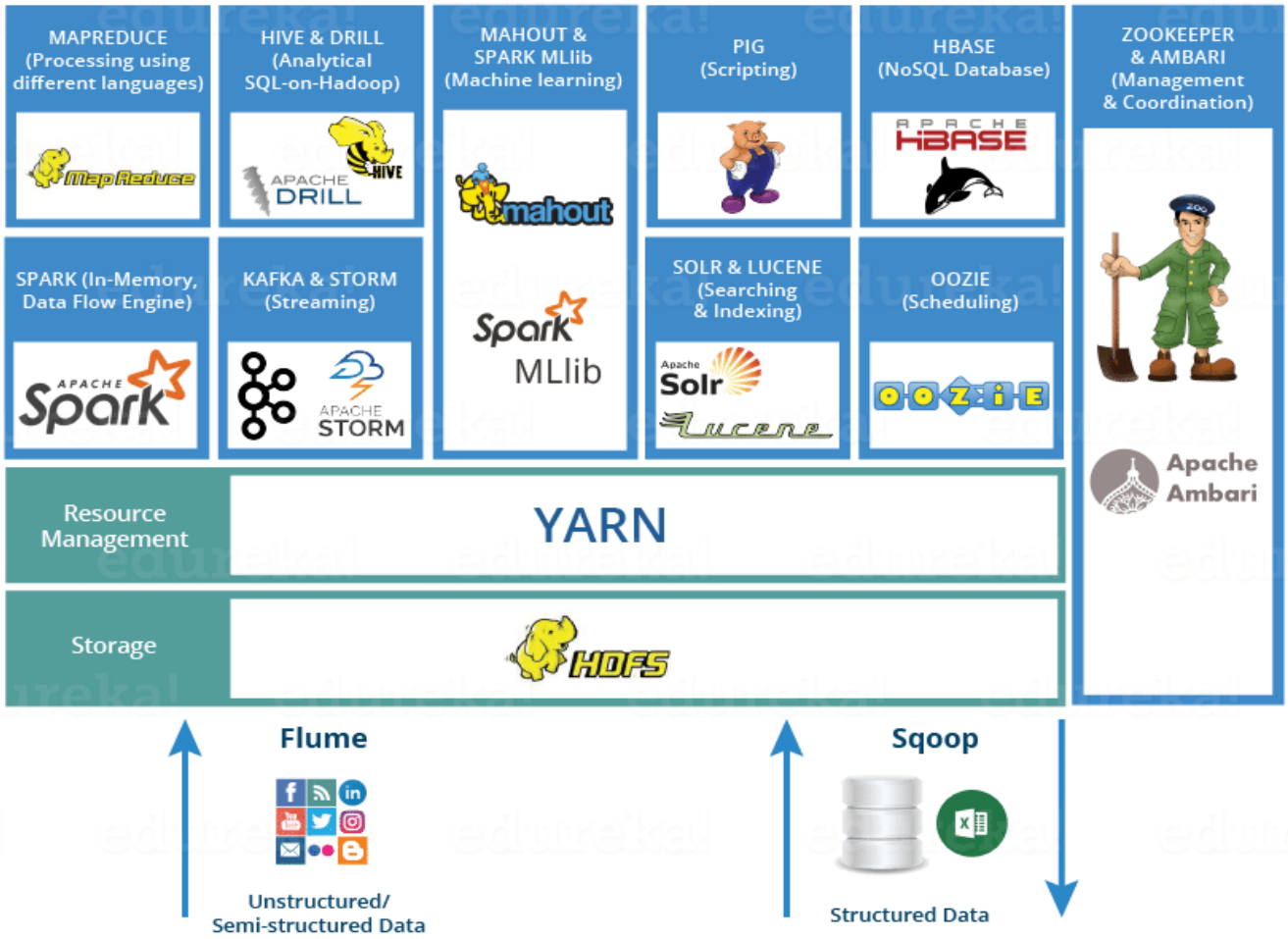
Main goal of stream mining: make sure that the constructed model is the most accurate and up-to-date.

Data Sufficiency

- **Definition:**
  - A dataset is considered “sufficient” if adding more data items will not increase the final accuracy of a trained model significantly.
  - It can be judged by experience.
- We normally do not know if a dataset is sufficient or not.
- **Sufficiency detection:**
  - Expensive “progressive sampling” experiment.
    - Keep on adding data and stop when accuracy doesn’t increase significantly.
- Dependent on both dataset and algorithm
  - Difficult to make a general claim

More about Big Data Tools

Hadoop Ecosystem



HDFS -> Hadoop Distributed File System
YARN -> Yet Another Resource Negotiator
MapReduce -> Data processing using programming
Spark -> In-memory Data Processing
PIG, HIVE-> Data Processing Services using Query (SQL-like)
HBase -> NoSQL Database

Mahout, Spark MLlib -> Machine Learning
Apache Drill -> SQL on Hadoop
Zookeeper -> Managing Cluster
Oozie -> Job Scheduling
Flume, Sqoop -> Data Ingesting Services
Solr & Lucene -> Searching & Indexing
Ambari -> Provision, Monitor and Maintain cluster

Introduction to Map Reduce

What is MapReduce?

- MapReduce is a programming model Google has used successfully is processing its “big-data” sets (~ 20000 peta bytes per day)
  - Users specify the computation in terms of a map and a reduce function,
  - Underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, clusters of commodity hardware in a reliable manner.
  - A processing technique and a program model for distributed computing based on java.
  - Underlying system also handles machine failures, efficient communications, and performance issues.

-- Reference: Dean, J. and Ghemawat, S. 2008. **MapReduce: simplified data processing on large clusters.** *Communication of ACM* 51, 1 (Jan. 2008), 107-113.

How MapReduce Works?

- The whole process goes through four phases of execution namely, splitting, mapping, shuffling, and reducing.
- Consider you have following input data for your Map Reduce Program

Input Data

Welcome to Hadoop Class
Hadoop is good
Hadoop is bad

The final output of the MapReduce task is

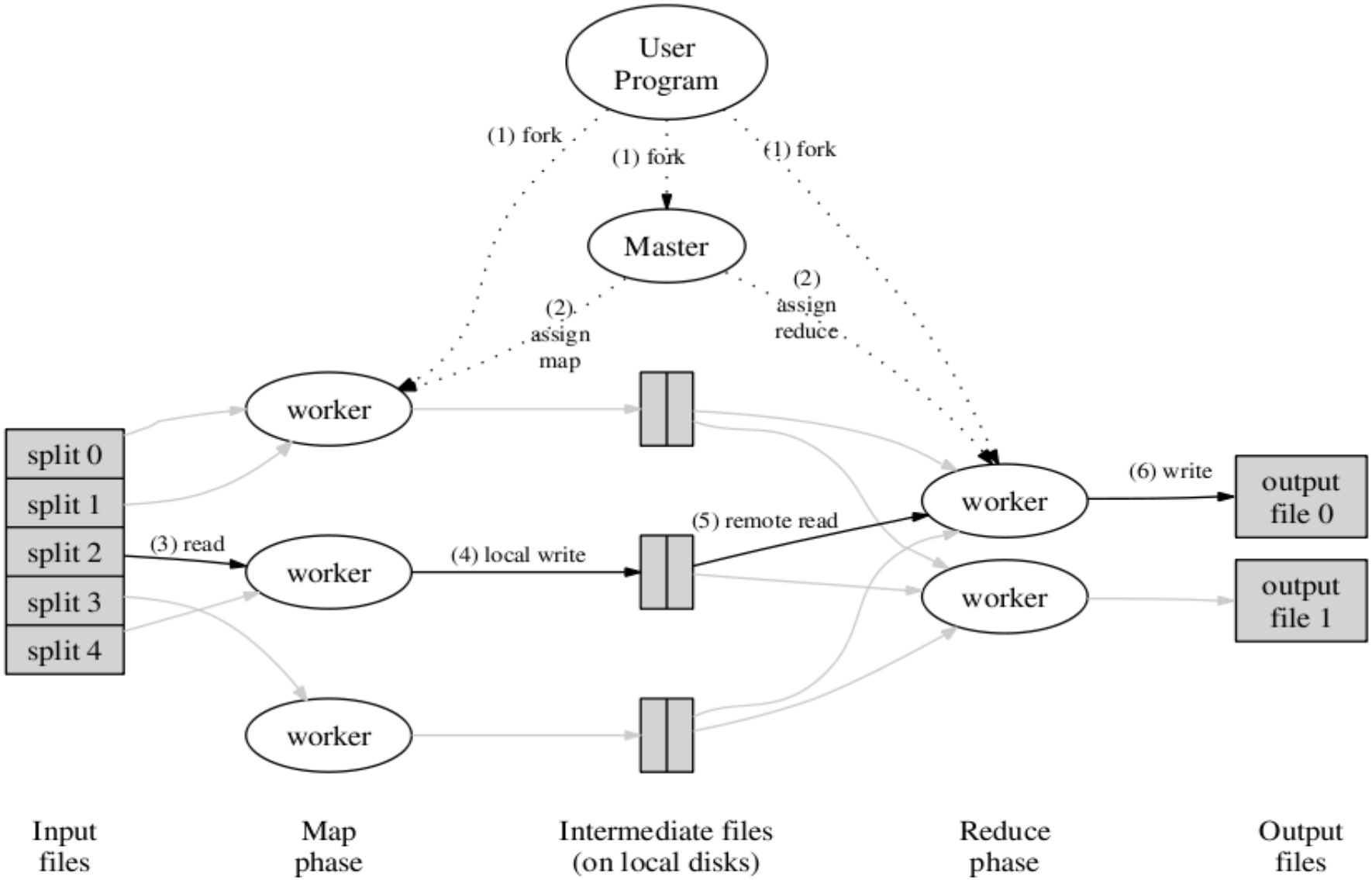
bad	1
Class	1
good	1
Hadoop	3
is	2
to	1
Welcome	1

MapReduce Architecture

- One map task (map function) is created for each split.
- It is always beneficial to have multiple splits because the time taken to process a split is small as compared to the time taken for processing of the whole input.
- When the splits are smaller, the processing is better to load balanced since we are processing the splits in parallel.
- However, it is also not desirable to have splits too small in size. When splits are too small, the overload of managing the splits and map task creation begins to dominate the total job execution time.
- For most jobs, it is better to make a split size equal to the size of an HDFS block (which is 64 MB, by default).

MapReduce Characteristics

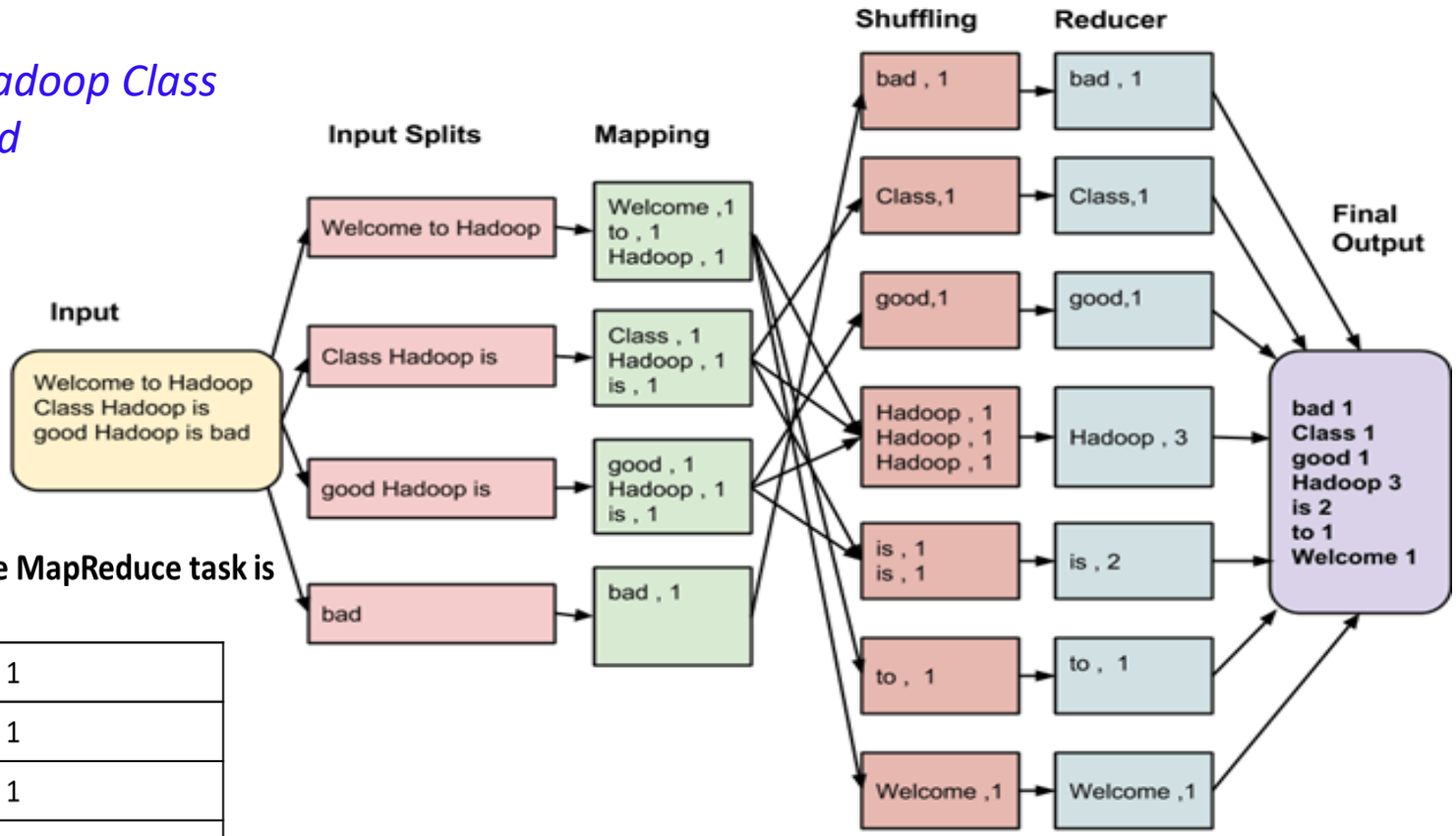
- Very large scale data: peta, exa bytes
- Write once and read many data: allows for parallelism without mutexes
- All the map should be completed before reduce operation starts.
- Map and reduce operations are typically performed by the same physical processor.
- Number of map tasks and reduce tasks are configurable.
- Commodity hardware and storage.
- Runtime takes care of splitting and moving data for operations.
- Special distributed file system. Example: Hadoop Distributed File System and Hadoop Runtime.


User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files

MapReduce Architecture

Input Data

Welcome to Hadoop Class
Hadoop is good
Hadoop is bad

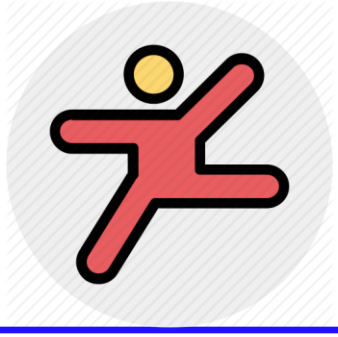


The final output of the MapReduce task is

bad	1
Class	1
good	1
Hadoop	3
is	2
to	1
Welcome	1

MapReduce Phases

- **Input Splits**
  - Job is divided into fixed-size pieces called input splits
- **Mapping**
  - In this phase data in each split is passed to a mapping function to produce output values. In our example, a job of mapping phase is to count a number of occurrences of each word from input splits
- **Shuffling**
  - This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubed together along with their respective frequency.
- **Reducing**
  - In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase summarizes the complete dataset.



Hands-on Exercise

Working with Hadoop

Map Reduce Programing

1. Word count application
2. Youtube Video Rating Application
3. Average Salary of Male and Female Employees
4. Sales Review

Introduction to Hadoop Distributed file system (HDFS)

DFS or Distributed File System

- Distributed File System talks about managing data, i.e. files or folders across multiple computers or servers.
- In other words, DFS is a file system that allows us to store data over multiple nodes or machines in a cluster and allows multiple users to access data.
- The only difference is that, in case of Distributed File System, you store data in multiple machines rather than single machine.
- Even though the files are stored across the network, DFS organizes, and displays data in such a manner that a user sitting on a machine will feel like all the data is stored in that very machine.

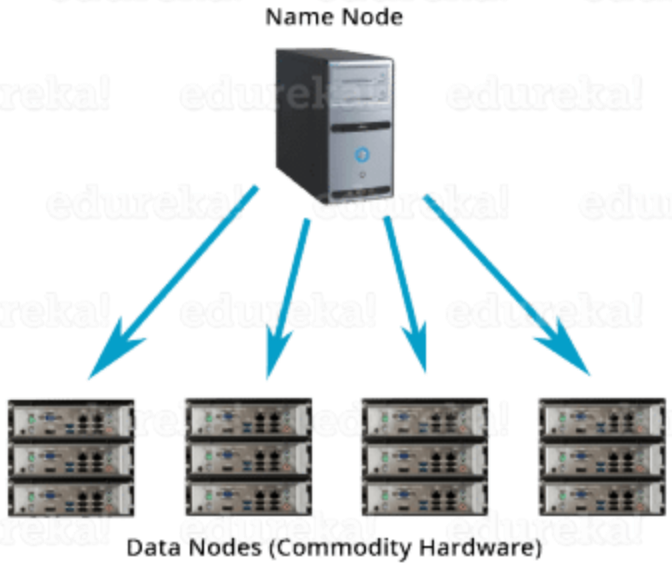
What is HDFS

- **Hadoop Distributed File System** or HDFS is a Java based distributed file system that allows you to store large data across multiple nodes in a Hadoop cluster.
- So, if you install Hadoop, you get HDFS as an underlying storage system for storing the data in the distributed environment.
- Example:
  - Imagine that you have ten machines or ten computers with a hard drive of 1 TB on each machine.
  - Now, HDFS says that if you install Hadoop as a platform on top of these ten machines, you will get HDFS as a storage service.
  - Hadoop Distributed File System is distributed in such a way that every machine contributes their individual storage for storing any kind of data.

Advantages of HDFS

Distributed Storage:

When you access Hadoop Distributed file system from any of the ten machines in the Hadoop cluster, you will feel as if you have logged into a single large machine which has a storage capacity of 10 TB (total storage over ten machines). What does it mean? It means that you can store a single large file of 10 TB which will be distributed over the ten machines (1 TB each). So, it is **not limited to the physical boundaries** of each individual machine.



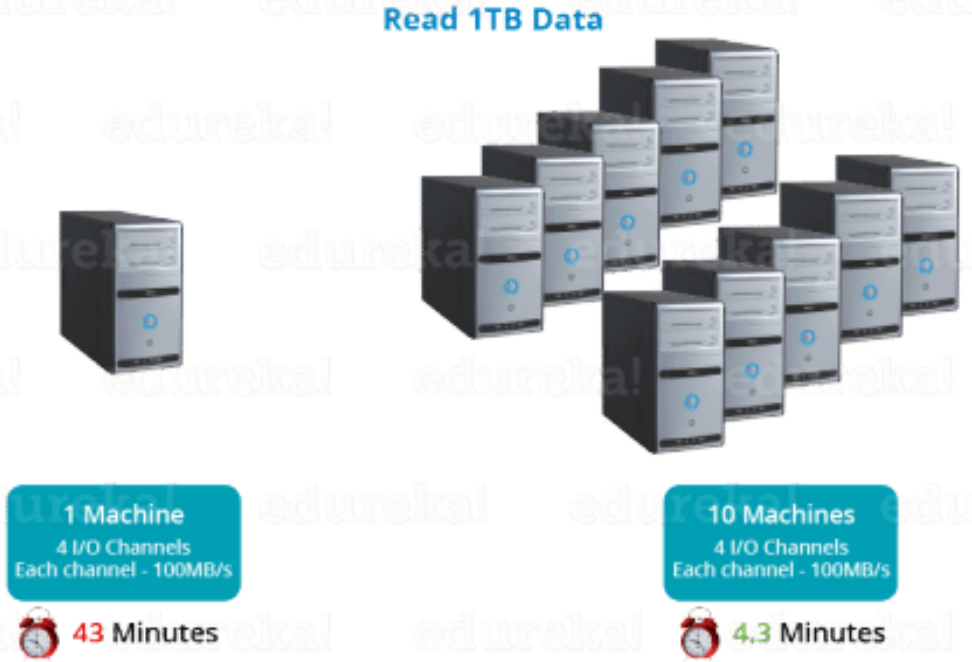
Advantages of HDFS

Distributed & Parallel Computation

- Because the data is divided across the machines, it allows us to take advantage of **Distributed and Parallel Computation**.
- Suppose, it takes 43 minutes to process 1 TB file on a single machine.
- How much time will it take to process the same 1 TB file when you have 10 machines in a Hadoop cluster with similar configuration –

43 minutes or 4.3 minutes?

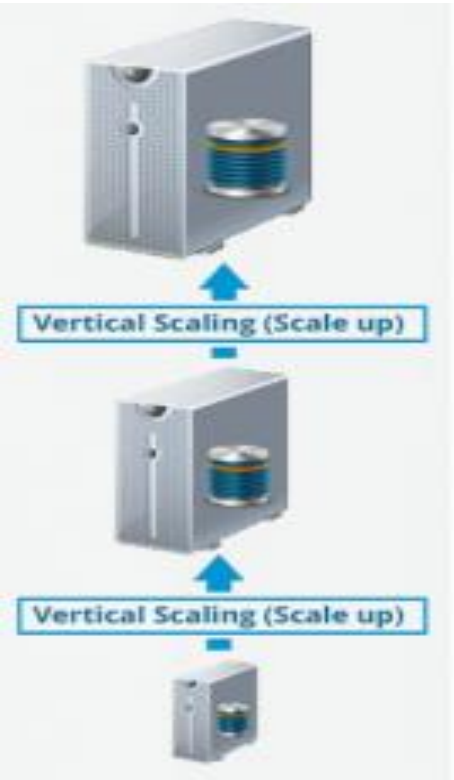
4.3 minutes



Advantages of HDFS

Scalability

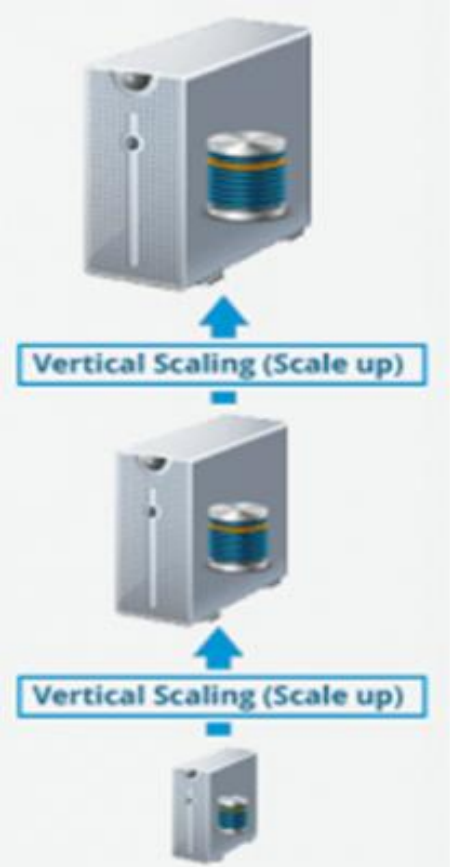
- There are two types of scaling: **vertical** and **horizontal.**
- In **vertical scaling (scale up),** you increase the hardware capacity of your system. In other words, you procure more RAM or CPU and add it to your existing system to make it more robust and powerful. But there are challenges associated with vertical scaling or scaling up:

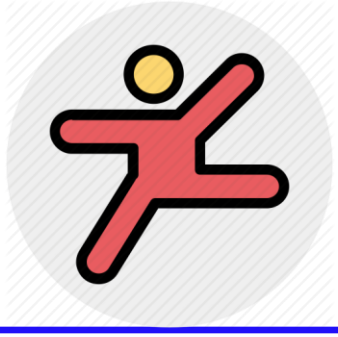


There is always a limit to which you can increase your hardware capacity. So, you can't keep on increasing the RAM or CPU of the machine.

Advantages of HDFS

- In case of **horizontal scaling (scale out),** you add more nodes to existing cluster instead of increasing the hardware capacity of individual machines. And most importantly, you can **add more machines on the go** i.e. Without stopping the system**.**





Hands-on Exercise

Working with HDFS

1.User Creation

2.Use of SSh Application in cluster

3.How to access HDFS

4.Read and create HDFS file

Introduction to Apache HIVE

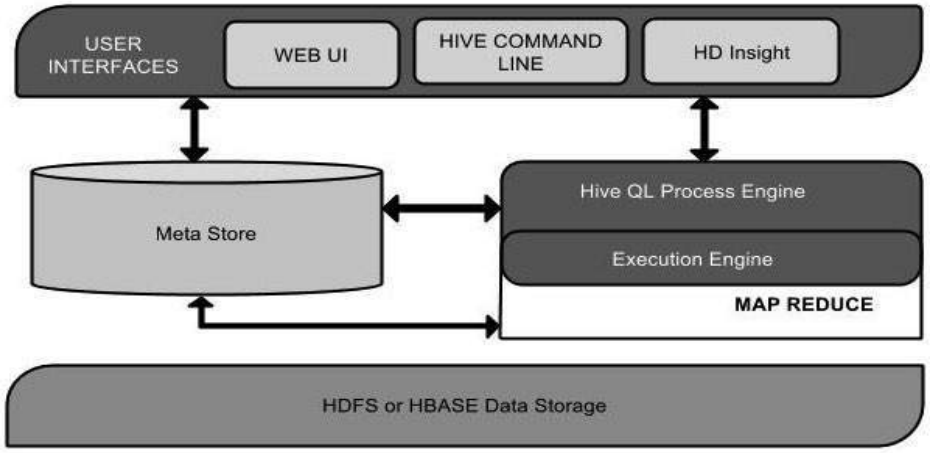
Apache HIVE

- It is a data warehouse and an ETL tool which provides an SQL-like interface between the user and the Hadoop distributed file system (HDFS) which integrates Hadoop.
- It is built on top of Hadoop.
- It is a software project that provides data query and analysis.
- It facilitates reading, writing and handling wide datasets that stored in distributed storage and queried by Structure Query Language (SQL) syntax.
- It is frequently used for data warehousing tasks like data encapsulation, Ad-hoc Queries, and analysis of huge datasets.
- It is designed to enhance scalability, extensibility, performance, fault-tolerance and loose-coupling with its input formats.

Story of Hive – from Facebook to Apache

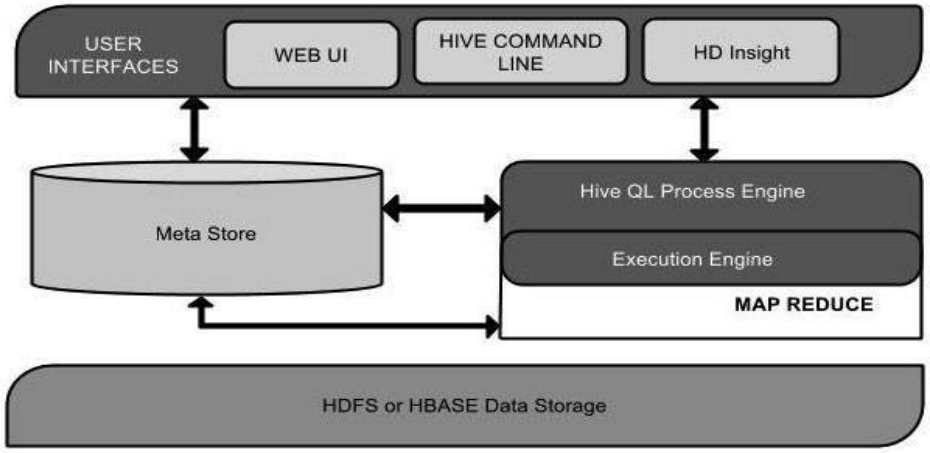
Apache HIVE Architecture

Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.



Apache HIVE Architecture

HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.



Modes of Hive

- **Local Mode –**
  - It is used, when the Hadoop is built under pseudo mode which have only one data node, when the data size is smaller in term of restricted to single local machine, and when processing will be faster on smaller datasets existing in the local machine.

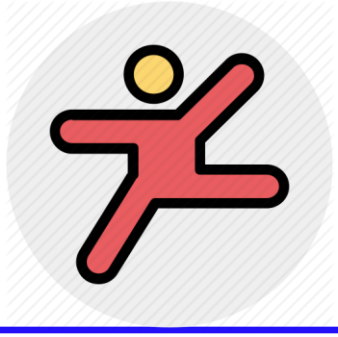
- **Map Reduce Mode –**
  - It is used, when Hadoop is built with multiple data nodes and data is divided across various nodes, it will function on huge datasets and query is executed parallelly, and to achieve enhanced performance in processing large datasets.

Benefits of Apache Hive

- **Ease of use** — Querying data is easy to learn with its SQL-like language.
- **Accelerated initial insertion of data —** Data does not have to be read, parsed, and serialized to a disk in the database's internal format, since Apache Hive reads the schema without checking the table type or schema definition. Compare this to a traditional database where data must be verified each time it is inserted.
- **Superior scalability, flexibility, and cost efficiency** — Apache Hive stores 100s of petabytes of data, since it stores data in the HDFS, making it a much more scalable solution than a traditional database. As a cloud-based Hadoop service, Apache Hive enables users to rapidly spin virtual servers up or down to accommodate fluctuating workloads.
- **Streamlined security** — Critical workloads can be replicated for disaster recovery.
- **Low overhead** — Insert-only tables have near-zero overhead. Since there is no renaming required, the solution is cloud friendly.
- **Exceptional working capacity** — Huge datasets support up to 100,000 queries/hour.

Limitations of Apache Hive

- Certain standard SQL functions, such as NOT IN, NOT LIKE, and NOT EQUAL, do not exist or require certain workarounds.
- Hive is not made for low-latency, real-time, or near-real-time querying.
- SQL queries translate to MapReduce, which means slower performance for certain queries compared to traditional RDBMS.



Hands-on Exercise

Woking with Hive

1.Create Table

2.Insert values into table

3.Other DDL and DML commands

Introduction to Flume

Apache Flume

- Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data.
- It has a simple and flexible architecture based on streaming data flows.
- It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms.
- It uses a simple extensible data model that allows for online analytic application.

Apache Flume with HDFS

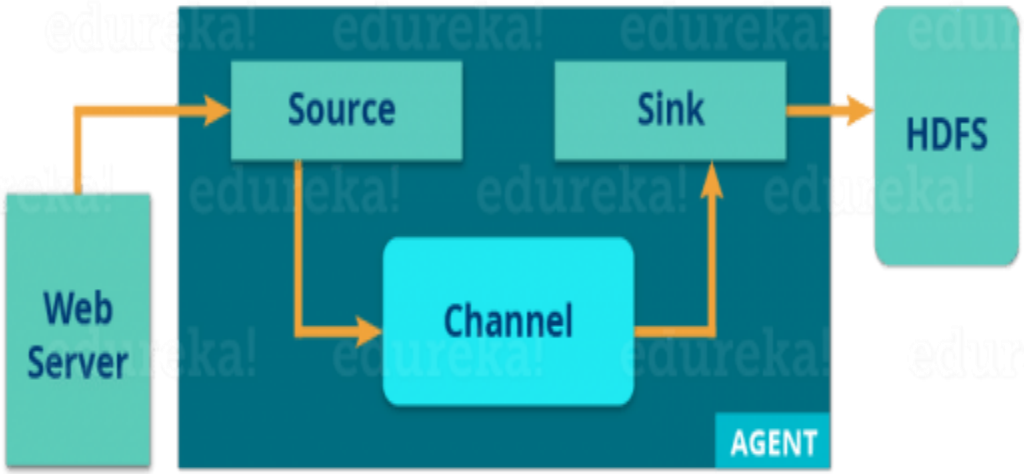
- Apache Flume is a tool for data ingestion in HDFS.
- It collects, aggregates and transports large amount of streaming data such as log files, events from various sources like network traffic, social media, email messages etc. to HDFS.
- The main idea behind the Flume's design is to capture streaming data from various web servers to HDFS. It has simple and flexible architecture based on streaming data flows.

Apache Flume with HDFS

- Apache Flume is a tool for data ingestion in HDFS.
- It collects, aggregates and transports large amount of streaming data such as log files, events from various sources like network traffic, social media, email messages etc. to HDFS.
- The main idea behind the Flume's design is to capture streaming data from various web servers to HDFS. It has simple and flexible architecture based on streaming data flows.

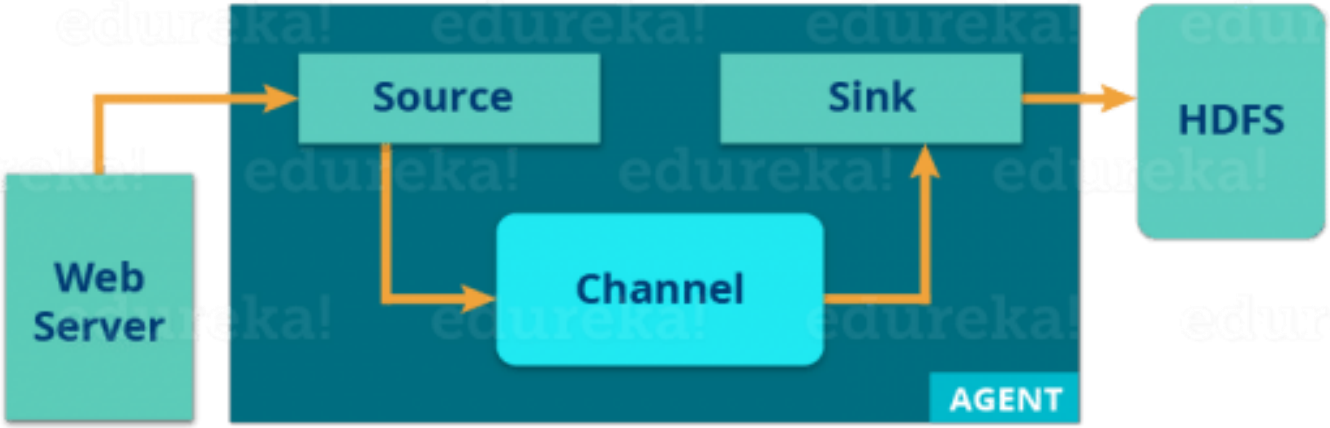
Flume Architecture

- There is a Flume agent which ingests the streaming data from various data sources to HDFS. From the diagram, you can easily understand that the web server indicates the data source. Twitter is among one of the famous sources for streaming data.
- The flume agent has 3 components: source, sink and channel.



Flume Architecture

- **Source:** It accepts the data from the incoming streamline and stores the data in the channel.
- **Channel**: In general, the reading speed is faster than the writing speed. Thus, we need some buffer to match the read & write speed difference. Basically, the buffer acts as a intermediary storage that stores the data being transferred temporarily and therefore prevents data loss. Similarly, channel acts as the local storage or a temporary storage between the source of data and persistent data in the HDFS.
- **Sink:** Then, our last component i.e. Sink, collects the data from the channel and commits or writes the data in the HDFS permanently.

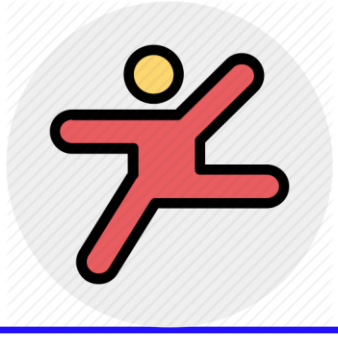


Advantages of Apache Flume

- Flume is scalable, reliable, fault tolerant and customizable for different sources and sinks.
- Apache Flume can store data in centralized stores (i.e data is supplied from a single store) like HBase & HDFS.
- Flume is horizontally scalable.
- If the read rate exceeds the write rate, Flume provides a steady flow of data between read and write operations.
- Flume provides reliable message delivery.
- The transactions in Flume are channel-based where two transactions (one sender & one receiver) are maintained for each message.

Advantages of Apache Flume

- Using Flume, we can ingest data from multiple servers into Hadoop.
- It gives us a solution which is reliable and distributed and helps us in collecting, aggregating and moving large amount of data sets like Facebook, Twitter and e-commerce websites.
- It helps us to ingest online streaming data from various sources like network traffic, social media, email messages, log files etc. in HDFS.
- It supports a large set of sources and destinations types.
- The architecture is one which is empowering Apache Flume with these benefits. Now, as we know the advantages of Apache Flume, lets move ahead and understand Apache Flume architecture.



Hands-on Exercise

Woking with Flume

- **Installation of flume**
- **Demonstration of telnet communication via console**
- **Application of flume on HDFS**

Introduction to Apache Mahout

Apache Mahout

- Apache Mahout is a machine-learning and data mining library. It provides three core features forprocessing large data sets.
  - **Clustering** is the ability to identify related documents to each other based on the content of each document.
  - **Classification** is the ability to categorize a document into an existing category based on the content of the document.
  - **Collaborative filtering** is the ability to provide filters that are based on the similarities and differences of tastes between users.

Mahout Use Cases

- Yahoo: Spam Detection
- Foursquare: Recommendations
- SpeedDate.com: Recommendations
- Adobe: User Targetting
- Amazon: Personalization Platform

Use case Example

- Predict what the user likes based on
  - His/Her historical behavior

Customers Who Bought This Item Also Bought

Pattern Recognition and Machine Learning... by Christopher M. Bishop

★★★★☆ (50)

$76.10

The Elements of Statistical Learning: Data Minin... by Trevor Hastie

★★★★☆ (38)

$71.96

Pattern Classification (2nd Edition) by Richard O. Duda

★★★½☆ (29)

$88.52

Features of Apache Mahout

- The algorithms of Mahout are written on top of Hadoop, so it works well in distributed environment. Mahout uses the Apache Hadoop library to scale effectively in the cloud.
- Mahout offers the coder a ready-to-use framework for doing data mining tasks on large volumes of data.
- Mahout lets applications to analyze large sets of data effectively and in quick time.
- Includes several MapReduce enabled clustering implementations such as kmeans, fuzzy k-means, Canopy, Dirichlet, and Mean-Shift.
- Supports Distributed Naive Bayes and Complementary Naive Bayes classification implementations.
- Comes with distributed fitness function capabilities for evolutionary programming.
- Includes matrix and vector libraries

Applications of Mahout

- Companies such as Adobe, Facebook, LinkedIn, Foursquare, Twitter, and Yahoo use Mahout internally.
- Foursquare helps you in finding out places, food, and entertainment available in a particular area. It uses the recommender engine of Mahout.
- Twitter uses Mahout for user interest modelling.
- Yahoo! uses Mahout for pattern mining.

Introduction to Elasticsearch

Basics of Elasticsearch

elasticsearch

Elasticsearch

- In today's IT world, a voluminous amount of data sizing approx 2.5 Quintillion bytes is generated every day.
- This data majorly comes from different sources, for example, social media sites, video sharing sites, and medium to large-scale organizations.
- This data is referred as data ocean or in more general terms called the Big Data.
- A considerable part of this data is insignificant, unstructured and scattered when it's alone.
- To make sense out of it you need analytic tools.

Elasticsearch

- Elasticsearch is a highly scalable open-source full-text search and analytics engine.
- It allows you to store, search, and analyze big volumes of data quickly and in near real time.
- It is generally used as the underlying engine/technology that powers applications that have complex search features and requirements.
- It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

Advantages of Elasticsearch

- **Scalability:** Elasticsearch is very easy to scale and reliable as well. It is a very important feature which helps to simplify the complex architectures and save time during the implementation of projects.
- **Speed:** Elasticsearch uses distributed inverted indices to find the best matches for your full-text searches. This makes it really fast even when searching from very large data sets.
- **Easy to use API**: Elasticsearch provides simple RESTful APIs and uses schema-free JSON documents which makes indexing, searching, and querying the data really easy.
- **Multilingual:** One of the most distinct features Elasticsearch has is, it is multilingual. It supports a wide variety of documents written in different languages like Arabic, Brazilian, Chinese, English, French, Hindi, Korean etc.

Advantages of Elasticsearch

- **Document-Oriented:** Elasticsearch stores real-world complex entities as structured JSON documents and indexes all fields by default to make the data searchable. Since there are no rows and columns of data, you can perform complex full-text search easily.
- **Auto-completion:** Elasticsearch also provides autocompletion functionality. By predicting the word using very few characters, autocompletion speeds up human-computer interaction.
- **Schema-Free:** Elasticsearch is schema-free as it accepts JSON documents. It tries to detect the data structure, index the data and thus makes the data searchable

Introduction to Kibana

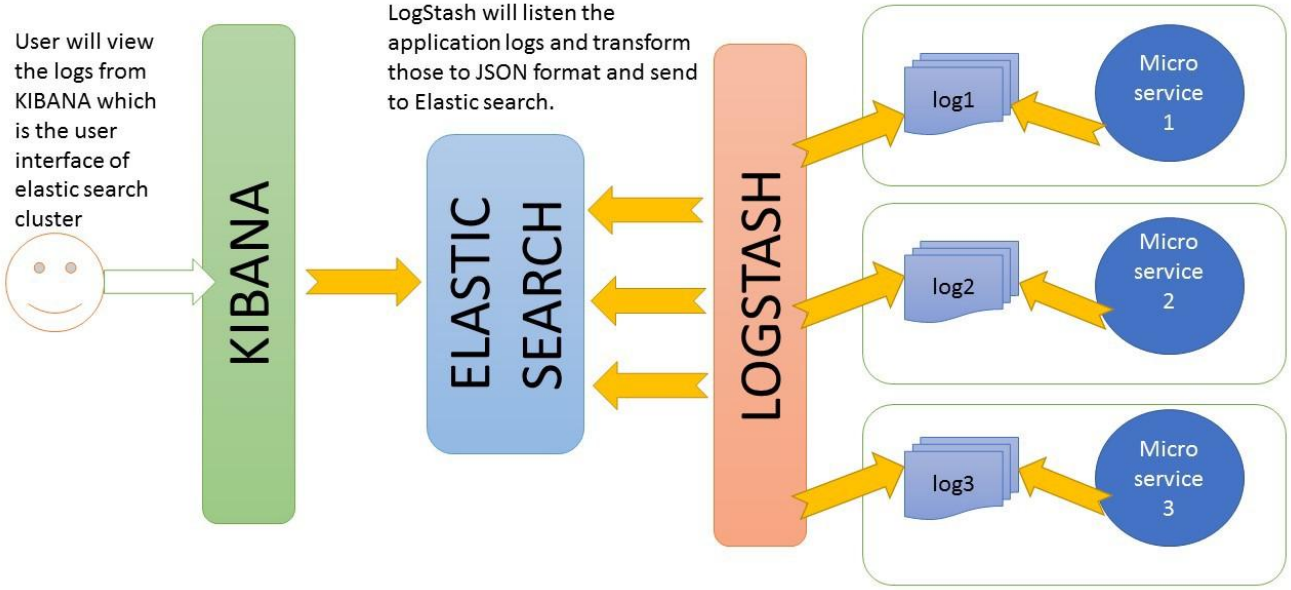
Basics of Kibana

Kibana

- Kibana is an open source analytics and visualization platform designed to work with Elasticsearch.
- Kibana can be used to search, view, and interact with data stored in Elasticsearch indices.
- We can easily perform advanced data analysis and visualize our data in a variety of charts, tables, and maps.
- Kibana makes it easy to understand large volumes of data. Its simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time.
- Setting up Kibana is a snap. You can install Kibana and start exploring your Elasticsearch indices in minutes — no code, no additional infrastructure required.

Kibana

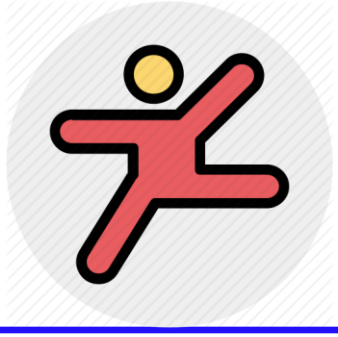
- Kibana also provides a presentation tool, referred to as Canvas, that allows users to create slide decks that pull live data directly from Elasticsearch.
- The combination of Elasticsearch, Logstash, and Kibana, referred to as the "Elastic Stack" (formerly the "ELK stack"), is available as a produ



ELK stack interaction with different applications based on Log file

Kibana Benefits

- **INTERACTIVE CHARTS**
  - Kibana offers intuitive charts and reports that you can use to interactively navigate through large amounts of log data. You can dynamically drag time windows, zoom in and out of specific data subsets, and drill down on reports to extract actionable insights from your data.
- **MAPPING SUPPORT**
  - Kibana comes with powerful geospatial capabilities so you can seamlessly layer in geographical information on top of your data and visualize results on maps.
- **PRE-BUILT AGGREGATIONS AND FILTERS**
  - Using Kibana's pre-built aggregations and filters, you can run a variety of analytics like histograms, top-N queries, and trends with just a few clicks.
- **EASILY ACCESSIBLE DASHBOARDS**
  - You can easily set up dashboards and reports and share them with others. All you need is a browser to view and explore the data.



Hands-on Exercise

1. Request to Elasticsearch and response to Kibana
2. Use of post and get methods
3. Search in database
4. Update
5. Bulk insertion

Introduction to Apache Kafka

Introduction to Apache Kafka

- Kafka® is used for building real-time data pipelines and streaming apps.
- It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies.
- A streaming platform has three key capabilities:
  - Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.
  - Store streams of records in a fault-tolerant durable way.
  - Process streams of records as they occur

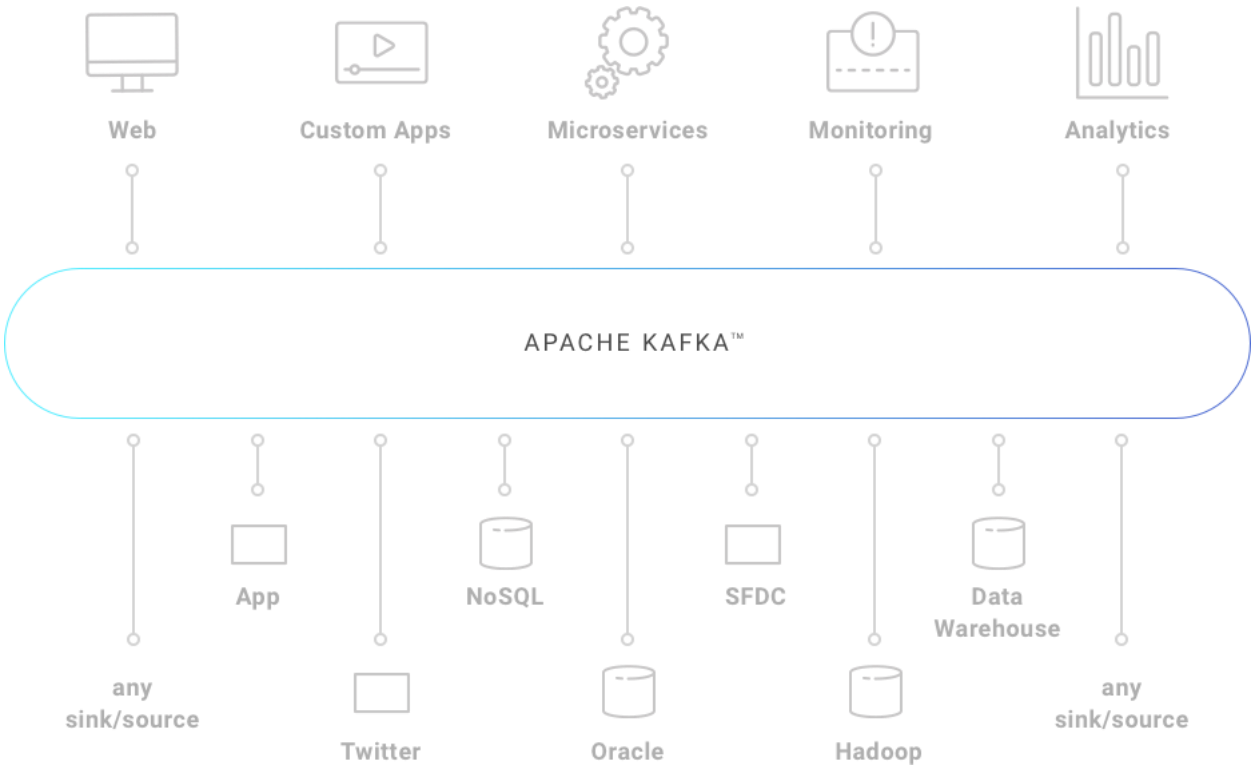
Introduction to Apache Kafka

- It is an apache project initially developed at LinkedIn
  - Distributed publish-subscribe messaging system
  - Designed for processing of real time activity stream data e.g. logs, metrics collections
  - Written in Scala
- Features
  - – Persistent messaging
  - – High-throughput
  - – Supports both queue and topic semantics
  - – Uses Zookeeper for forming a cluster of nodes (producer/consumer/broker)
  - and many more…
  - http://kafka.apache.org/

Introduction to Apache Kafka

- Kafka is generally used for two broad classes of applications:
  - Building real-time streaming data pipelines that reliably get data between systems or applications
  - Building real-time streaming applications that transform or react to the streams of data
- First a few concepts:
  - Kafka is run as a cluster on one or more servers that can span multiple datacenters.
  - The Kafka cluster stores streams of *records* in categories called *topics*.
  - Each record consists of a key, a value, and a timestamp.

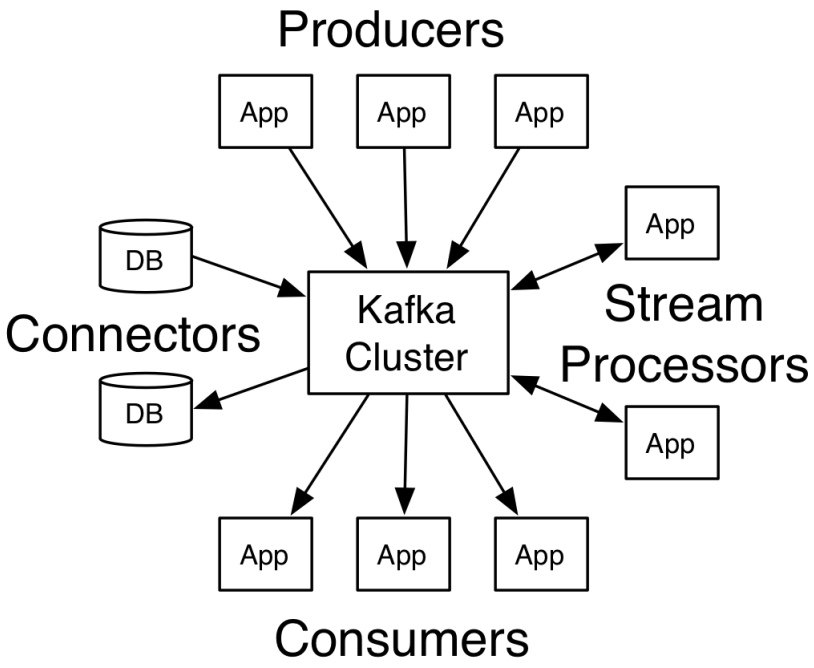
Where Apache Kafka Fits In



Introduction to Apache Kafka

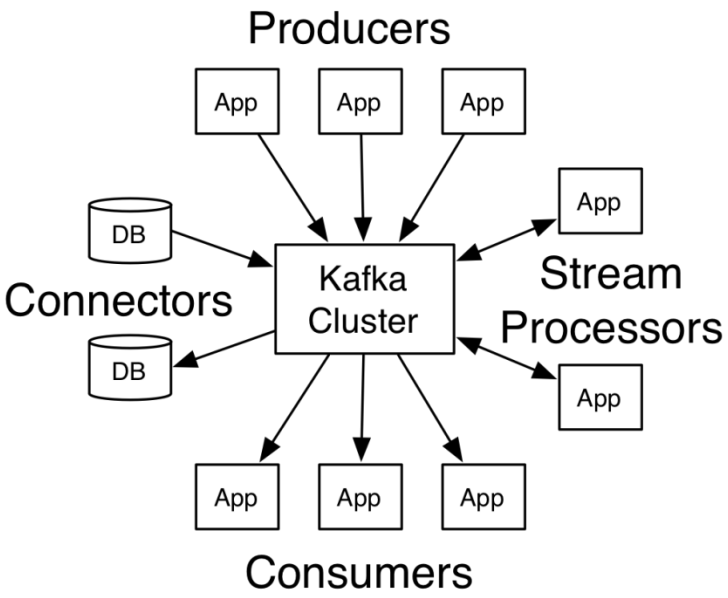
Kafka has four core APIs:

- **The Producer API** allows an application to publish a stream of records to one or more Kafka topics.
- **The Consumer API** allows an application to subscribe to one or more topics and process the stream of records produced to them.



Introduction to Apache Kafka

- **The Streams API** allows an application to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.
- **The Connector API** allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.



How Can Apache Kafka Help You?

- **Publish + Subscribe**
  - Using its immutable commit log, you can subscribe to it, and publish data to any number of systems or real-time applications.
  - Unlike messaging queues, Kafka is a highly scalable, fault tolerant distributed system, allowing it to be deployed for applications like managing passenger and driver matching at Uber.
  - It is providing real-time analytics and predictive maintenance for British Gas' smart home, and performing numerous real-time services across all of LinkedIn.
  - This unique performance makes it perfect to scale from one app to company-wide use.

How Can Apache Kafka Help You?

Store

- An abstraction of a distributed commit log commonly found in distributed databases, Apache Kafka provides durable storage.
- Kafka can act as a ‘source of truth’, being able to distribute data across multiple nodes for a highly available deployment within a single data center or across multiple availability zones.

How Can Apache Kafka Help You?

Process

- An event streaming platform would not be complete without the ability to manipulate that data as it arrives.
- The Streams API within Apache Kafka is a powerful, lightweight library that allows for on-the-fly processing, letting you aggregate, create windowing parameters, perform joins of data within a stream, and more.
- Perhaps best of all, it is built as a Java application on top of Kafka, keeping your workflow intact with no extra clusters to maintain.
- Kafka is very fast, performs 2 million writes/sec.

Introduction to Apache Zookeeper

Apache ZooKeeper™

What is a Distributed System

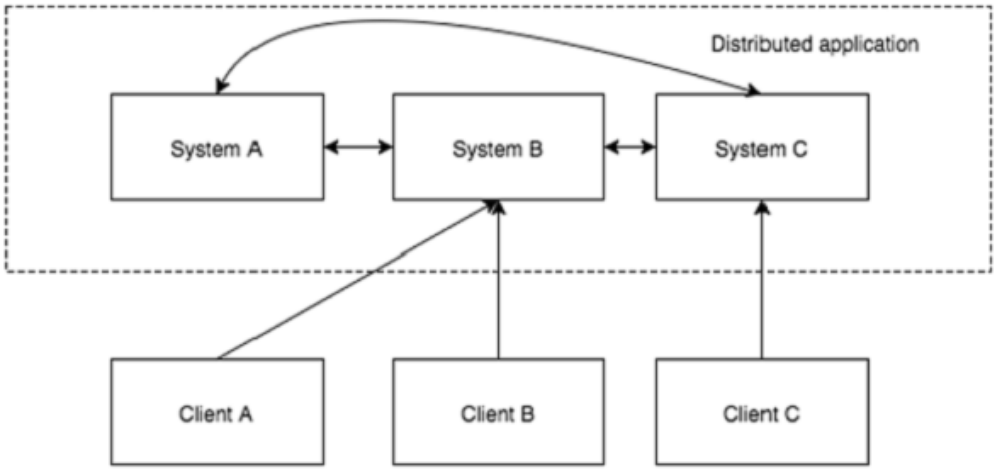
- “A Distributed system consists of multiple computers that communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. ”

  - Wikipedia

- Normally, complex and time-consuming tasks, which will take hours to complete by a non-distributed application (running in a single system) can be done in minutes by a distributed application by using computing capabilities of all the system involved.

What is a Distributed System

- A distributed application has two parts, Server and Client application. Server applications are actually distributed and have a common interface so that clients can connect to any server in the cluster and get the same result.
- Client applications are the tools to interact with a distributed application.



Benefits of Distributed Applications

- **Reliability** − Failure of a single or a few systems does not make the whole system to fail.
- **Scalability** − Performance can be increased as and when needed by adding more machines with minor change in the configuration of the application with no downtime.
- **Transparency** − Hides the complexity of the system and shows itself as a single entity / application.

Challenges of Distributed Applications

- **Race condition** – Two or more machines trying to perform a particular task, which actually needs to be done only by a single machine at any given time. For example, shared resources should only be modified by a single machine at any given time.
- **Deadlock** – Two or more operations waiting for each other to complete indefinitely.
- **Inconsistency** – Partial failure of data.

What is Apache ZooKeeper Meant For?

- Apache ZooKeeper is a service used by a cluster (group of nodes) to coordinate between themselves and maintain shared data with robust synchronization techniques.
- ZooKeeper is itself a distributed application providing services for writing a distributed application.
- The common services provided by ZooKeeper are as follows –
  - **Naming service –** Identifying the nodes in a cluster by name. It is similar to DNS, but for nodes.
  - **Configuration management –** Latest and up-to-date configuration information of the system for a joining node.
  - **Cluster management –** Joining / leaving of a node in a cluster and node status at real time.
  - **Leader election –** Electing a node as leader for coordination purpose.
  - **Locking and synchronization service –** Locking the data while modifying it. This mechanism helps you in automatic fail recovery while connecting other distributed applications like Apache HBase.

Benefits of ZooKeeper

- Simple distributed coordination process- Race condition and deadlock are handled using fail-safe synchronization approach. Another main drawback is inconsistency of data, which ZooKeeper resolves with atomicity.
- Synchronization − Mutual exclusion and co-operation between server processes. This process helps in Apache HBase for configuration management.
- Serialization − Encode the data according to specific rules. Ensure your application runs consistently. This approach can be used in MapReduce to coordinate queue to execute running threads.
- Atomicity − Data transfer either succeed or fail completely, but no transaction is partial. Inconsistency of data, can also be resolved using Zookeeper.

Why is Zookeeper necessary for Apache Kafka?

- **Controller election**
  - Whenever a node shuts down, a new controller can be elected and it can also be made sure that at any given time, there is only one controller and all the follower nodes have agreed on that.
- **Configuration of Topics**
  - The configuration regarding all the topics including the list of existing topics, the number of partitions for each topic, the location of all the replicas, list of configuration overrides for all topics and which node is the preferred leader, etc.
- **Access control lists**
  - Access control lists or ACLs for all the topics are also maintained within Zookeeper.
- **Membership of the cluster**
  - Zookeeper also maintains a list of all the brokers that are functioning at any given moment and are a part of the cluster.

References

- Official Websites of Apache Software

Thank you