Introduction to Big Data

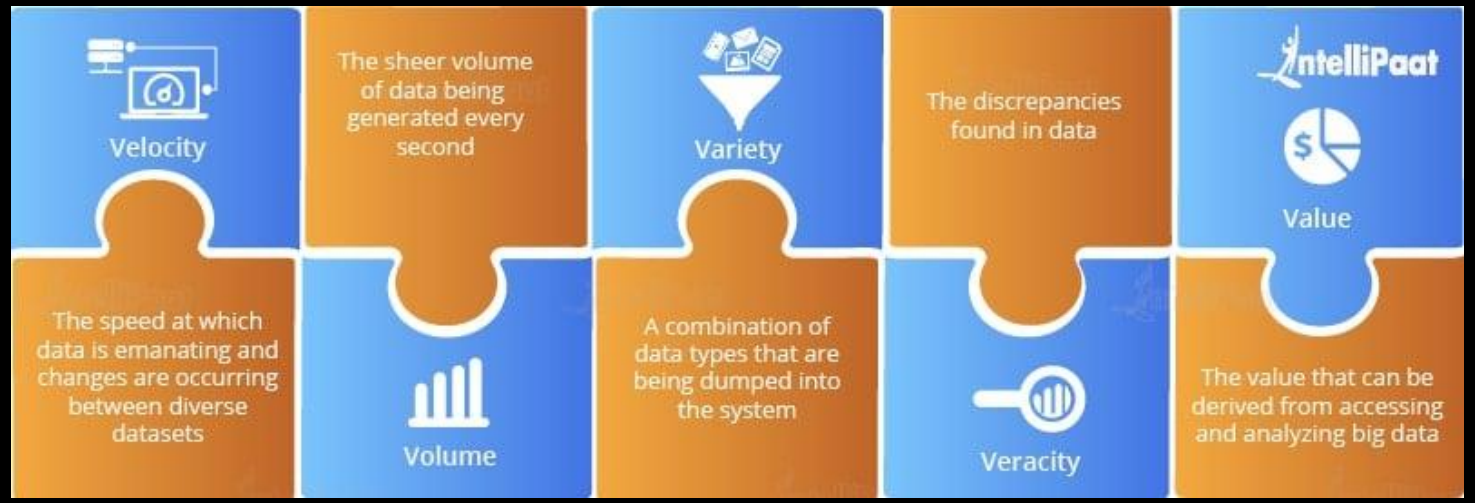
Unit 5

Introduction to Big Data

- Big data refers to a process that is used when traditional data mining and handling techniques isn't capable or sufficient enough uncover the insights and meaning of the underlying data.
- Collection of data sets so large and complex that it becomes difficult to process using on hand database management tools or traditional data processing applications.
- Big data is high-volume, high velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.

5 V's of Big Data

- Volume - Data at rest (too big)
- Variety - Data in many forms (too complex)
- Velocity -Data in motion(too fast)
- Veracity - Data in doubt(uncertainty)
- Value - Data into money



The Five V's of Big Data

Scale of Data

This refers to the sheer volume of data being generated every second.

6 Billion People have cell phones

40 Zettabytes of data will be created by 2020 and increase of 300 times from 2005

Most companies in the U.S. have at least **100 Terabytes** of data stored.

Analysis of Streaming Data

The New York Stock Exchange capture **1 TB of Trade Information**

Denotes the speed at which data is emanating and changes are occurring between the diverse data sets.

By 2016 it is projected there will be **18.9 Billion** network connections

Modern cars have close to **100 Sensors**

Uncertainty Of Data

1 in 3 Business leaders don't trust the information they use to make decisions

This refers to the discrepancies found in the data.

Poor data quality costa the US economy around **$ 3.1 Trillion a year**

Diffrent forms of data

4 Billion+ hours of video are watched on You Tube each month

30 Billion pieces of content are shared on facebook every month

400 Million tweets are sent per day by about 200 million monthly active users

As more and more data is being digitized.

Value Of Data

Having access to big data is all well and good but that's only useful if we can turn it into a value.

Volume: Scale of Data

- Refers to the vast amounts of data generated every second.
- This is where Big Data largely gets its name due to the sheer size of the data being collected.
- The actual size will vary based on the data being collected.
- For example, the user analytics of the Netflix database will be astronomical compared to e-commerce data for a small business, but both could be considered Big Data as it is a large amount of data which is being collected.
- We are not talking Terabytes but Brontobytes or Geopbytes.
- If we take all the data generated in the world between the beginning of time and 2008, the same amount of data will soon be generated every minute.

Variety: Different Forms of Data

- This refers to the different types of data we can now use.
- In the past we focused on structured data that fits neatly into tables or relational databases, such as financial data.
- In fact, 80% of the world's data is unstructured (text, images, video, voice, etc.)
- Big data technology means we can now analyze and bring together data of different types such as messages, social media conversations, photos, sensor data, video or voice recordings.

Velocity: Analysis of Streaming Data

- Refers to the speed at which new data is generated and the speed at which data moves around.
- Just think of social media messages going viral in seconds.
- Technology allows us now to analyze the data while it is being generated (in-memory analytics), without ever putting it into databases.
- Example: Google receives over 63,000 searches per second on any given day.

Veracity: Uncertainty of Data

- Veracity is the quality or trustworthiness of the data.
- With many forms of big data, quality and accuracy are less controllable.
- Big data and analytics technology now allows us to work with these type of data.
- There is little point to collecting Big Data if you are not confident that the resulting analyze can be trusted.
- For example, if you are piping all order data in but also including fraudulent or cancelled orders, you can’t trust the analysis of the e-commerce conversion rate because it will be artificially inflated.

Value: Turning Big Data into Value

- Having access to big data is no good unless we can turn it into value.
- Companies are starting to generate amazing value from their big data.

Scope of Big Data

- Increasing demand for Data Analytics
- Increasing enterprise adoption of Big Data
- Big Data finds application across various parallels of the industry
- Huge Job Opportunities & Meeting the Skill Gap
- Promises exponential salary growth
- Key Decision-Making Power

Challenges of Handling Big Data

- Lack of understanding of Big Data
- Dealing with data growth
- Confusion while Big Data tool selection
- Generating insights in a timely manner
- Recruiting and retaining big data talent
- Integrating disparate data sources
- Securing big data
- Organizational resistance

Commonly used tools for big data

- Hadoop
- Map-reduce programming
- HDFS
- Spark
- Apache Hive
- Apache Pig
- Apache Kafka
- Hbase
- NoSQL Database: MongoDB, Hbase, Cassandra etc.

Hadoop

- The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.
- It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.
- Rather than relying on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Hadoop (contd.)

The project includes these modules:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

HDFS – Hadoop File System

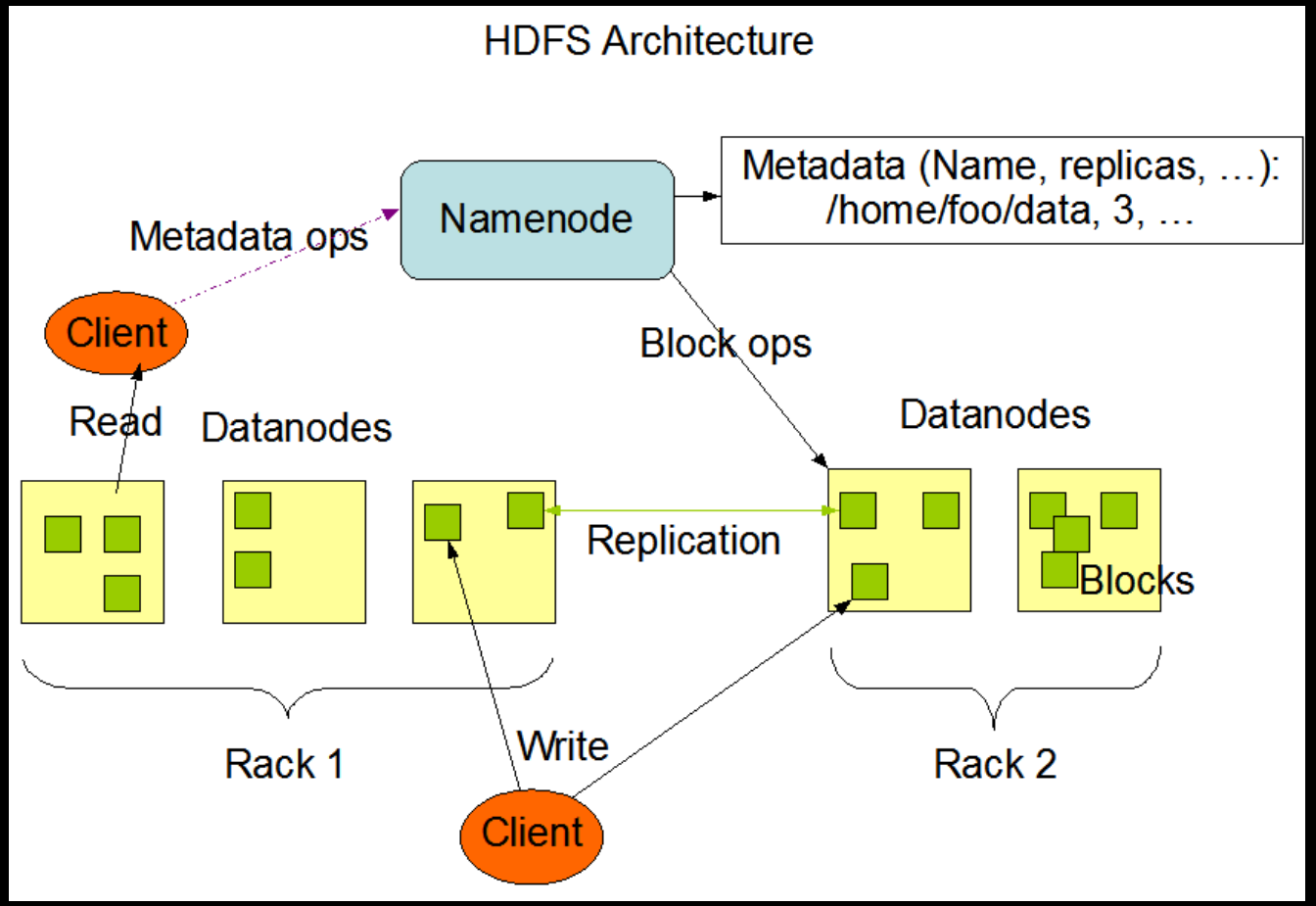
- The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware.
- It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant.
- HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware.
- HDFS provides high throughput access to application data and is suitable for applications that have large data sets.

HDFS Architecture Guide (apache.org)

HDFS (contd.)

- HDFS has a master/slave architecture.
- An HDFS cluster consists of a single Name Node, a master server that manages the file system namespace and regulates access to files by clients.
- In addition, there are a number of Data Nodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.

HDFS (contd.)


HDFS Architecture
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

HDFS (contd.)

- The Name Node and Data Node are pieces of software designed to run on commodity machines.
- These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the Name Node or the Data Node software.
- Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines.
- A typical deployment has a dedicated machine that runs only the Name Node software. Each of the other machines in the cluster runs one instance of the Data Node software.

Map reduce programming

- MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.
- MapReduce is a processing technique and a program model for distributed computing based on java.
- The MapReduce algorithm contains two important tasks, namely Map and Reduce.
- Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

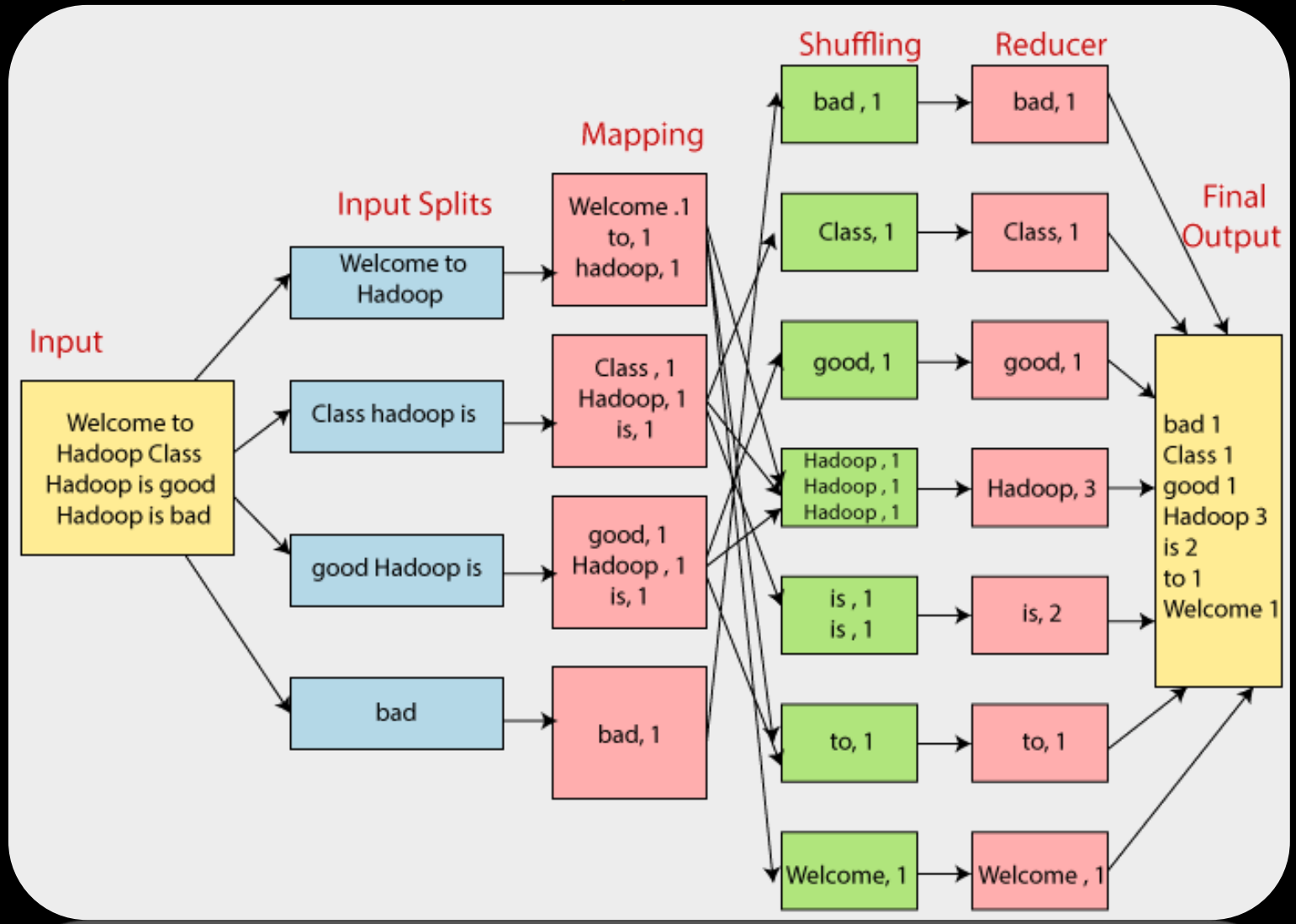
Map reduce programming (contd.)

- Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples.
- As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.
- The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes.
- Under the MapReduce model, the data processing primitives are called mappers and reducers.

Map reduce programming (contd.)

- Decomposing a data processing application into mappers and reducers is sometimes nontrivial.
- But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change.
- This simple scalability is what has attracted many programmers to use the MapReduce model.

Map Reduce - Example



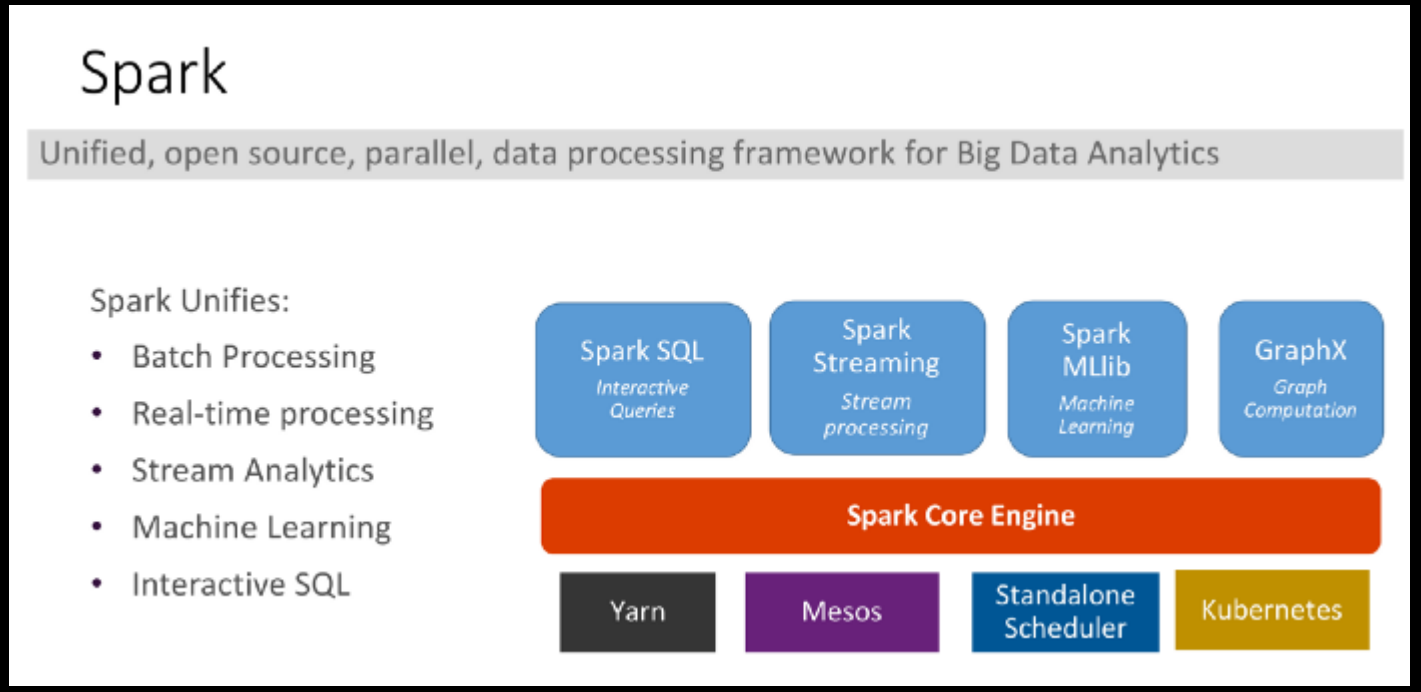
Spark

- Apache Spark is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.
- Apache Spark is a unified analytics engine for large-scale data processing.
- It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs.
- It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, pandas API on Spark for pandas workloads, MLlib for machine learning, GraphX for graph processing, and Structured Streaming for incremental computation and stream processing.

Key features of Spark

1. Batch/streaming data: Unify the processing of your data in batches and real-time streaming, using your preferred language: Python, SQL, Scala, Java or R.
2. SQL Analytics: Execute fast, distributed ANSI SQL queries for dashboarding and ad-hoc reporting. Runs faster than most data warehouses.
3. Data Science at scale: Perform Exploratory Data Analysis (EDA) on petabyte-scale data without having to resort to downsampling.
4. Machine Learning: Train machine learning algorithms on a laptop and use the same code to scale to fault-tolerant clusters of thousands of machines.

Spark Architecture



Spark Architecture (contd.)

Spark SQL and DataFrame

Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrame and can also act as distributed SQL query engine.

Streaming

Running on top of Spark, the streaming feature in Apache Spark enables powerful interactive and analytical applications across both streaming and historical data, while inheriting Spark's ease of use and fault tolerance characteristics.

Spark Architecture (contd.)

Mlib

Built on top of Spark, MLib is a scalable machine learning library that provides a uniform set of high-level APIs that help users create and tune practical machine learning pipelines.

```
# Every record of this DataFrame contains the label and
# features represented by a vector.
df = sqlContext.createDataFrame(data, ["label", "features"])

# Set parameters for the algorithm.
# Here, we limit the number of iterations to 10.
lr = LogisticRegression(maxIter=10)

# Fit the model to the data.
model = lr.fit(df)

# Given a dataset, predict each point's label, and show the results.
model.transform(df).show()
```

Spark Architecture (contd.)

Spark Core

Spark Core is the underlying general execution engine for the Spark platform that all other functionality is built on top of. It provides an RDD (Resilient Distributed Dataset) and in-memory computing capabilities.

Spark Sample code

```
# Creates a DataFrame based on a table named "people"
# stored in a MySQL database.
url = \
  "jdbc:mysql://yourIP:yourPort/test?user=yourUsername;password=yourPassword"
df = sqlContext \
  .read \
  .format("jdbc") \
  .option("url", url) \
  .option("dbtable", "people") \
  .load()

# Looks the schema of this DataFrame.
df.printSchema()

# Counts people by age
countsByAge = df.groupBy("age").count()
countsByAge.show()

# Saves countsByAge to S3 in the JSON format.
countsByAge.write.format("json").save("s3a://...")
```

Python code using spark

Hive

- Apache Hive is a distributed, fault-tolerant data warehouse system that enables analytics at a massive scale.
- It is a data warehouse software that facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.
- Structure can be projected onto data already in storage.
- A command line tool and JDBC driver are provided to connect users to Hive.
- Hive provides standard SQL functionality, including many of the later SQL:2003, SQL:2011, and SQL:2016 features for analytics.
- Hive's SQL can also be extended with user code via user defined functions (UDFs), user defined aggregates (UDAFs), and user defined table functions (UDTFs).

Hive (contd.)

- Built on top of Apache Hadoo, Hive provides the following features:
  - Tools to enable easy access to data via SQL, thus enabling data warehousing tasks such as extract/transform/load (ETL), reporting, and data analysis.
  - A mechanism to impose structure on a variety of data formats
  - Access to files stored either directly in Apache HDFS or in other data storage systems such as Apache HBase
  - Query execution via Apache Tez™, Apache Spark, or MapReduce
  - Procedural language with HPL-SQL

Hive (contd.)

- There is not a single "Hive format" in which data must be stored. Hive comes with built in connectors for comma and tab-separated values (CSV/TSV) text files, Apache Parquet, Apache ORC, and other formats.U
- Users can also extend Hive with connectors for other formats.
- Hive is not designed for online transaction processing (OLTP) workloads.
- It is best used for traditional data warehousing tasks.
- Hive is designed to maximize scalability (scale out with more machines added dynamically to the Hadoop cluster), performance, extensibility, fault-tolerance, and loose-coupling with its input formats.

Hive (contd.)

Creating Hive Tables

```
hive> CREATE TABLE pokes (foo INT, bar STRING);
```

creates a table called pokes with two columns, the first being an integer and the other a string.

```
hive> CREATE TABLE invites (foo INT, bar STRING) PARTITIONED BY (ds STRING);
```

creates a table called invites with two columns and a partition column called ds. The partition column is a virtual column. It is not part of the data itself but is derived from the partition that a particular dataset is loaded into.

Browsing through Tables

```
hive> SHOW TABLES;
```

lists all the tables.

```
hive> SHOW TABLES '.*s';
```

lists all the table that end with 's'. The pattern matching follows Java regular expressions.

```
hive> DESCRIBE invites;
```

shows the list of columns.

Real time Analytics with Apache Kafka

- Apache Kafka is the most popular open-source stream-processing software for collecting, processing, storing, and analyzing data at scale.
- Most known for its excellent performance, low latency, fault tolerance, and high throughput, it's capable of handling thousands of messages per second.

Apache Kafka (contd.)

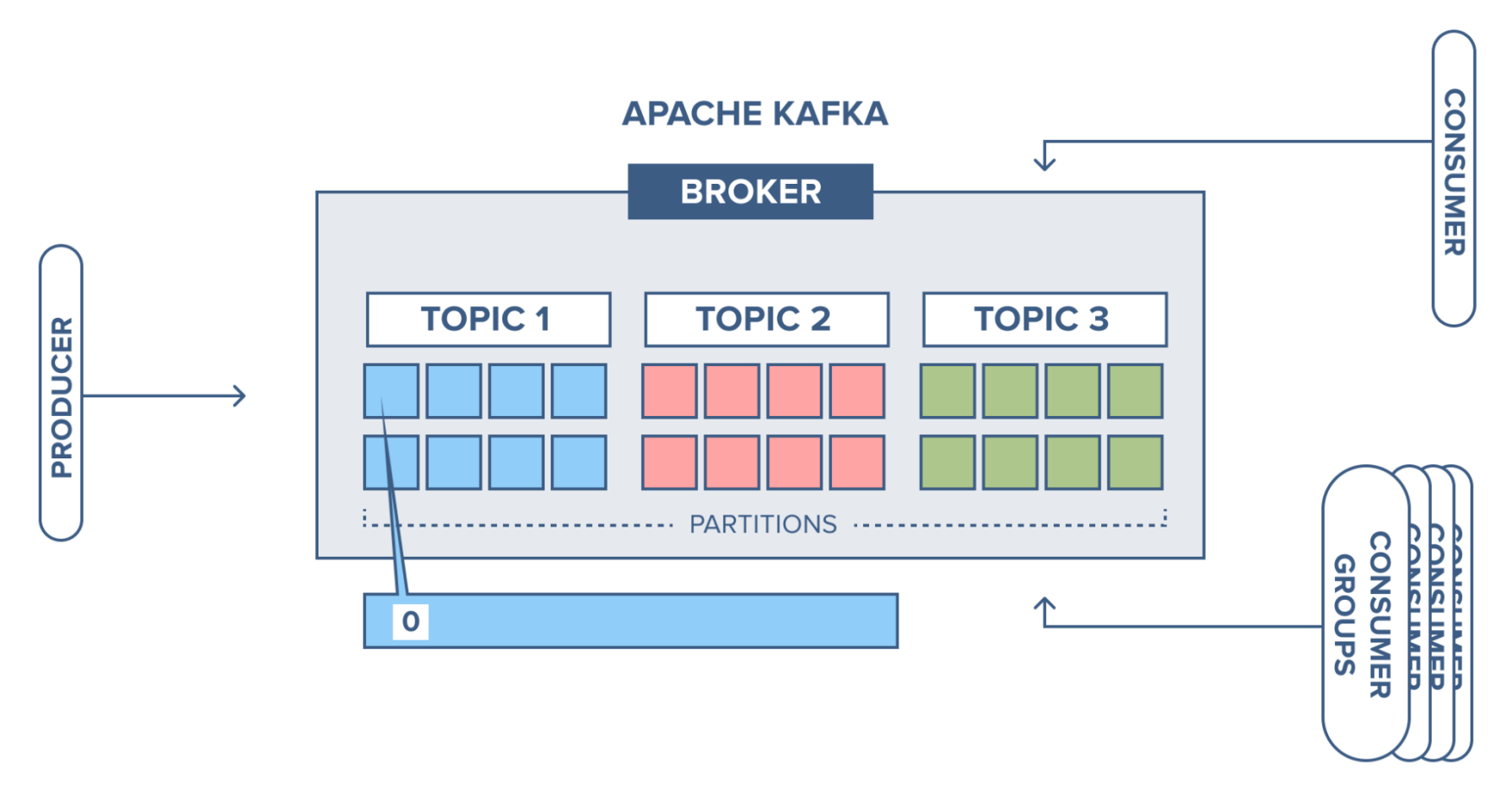
- Event streaming
  - is the practice of capturing data in real-time from event sources like databases, sensors, mobile devices, cloud services, and software applications in the form of streams of events;
  - storing these event streams durably for later retrieval; manipulating, processing, and reacting to the event streams in real-time as well as retrospectively;
  - and routing the event streams to different destination technologies as needed.
- Event streaming thus ensures a continuous flow and interpretation of data so that the right information is at the right place, at the right time.

Apache Kafka (contd.)

Kafka combines three key capabilities so we can implement our use cases for event streaming end-to-end :

1. To publish (write) and subscribe to (read) streams of events, including continuous import/export of your data from other systems.
2. To store streams of events durably and reliably for as long as you want.
3. To process streams of events as they occur or retrospectively.

Apache Kafka (contd.)



Apache Kafka (contd.)

- And all this functionality is provided in a distributed, highly scalable, elastic, fault-tolerant, and secure manner.
- Kafka can be deployed on bare-metal hardware, virtual machines, and containers, and on-premises as well as in the cloud.

End of the Chapter