BIG DATA ANALYSIS USING SPARK - R

27TH MAY 2017

AGENDA

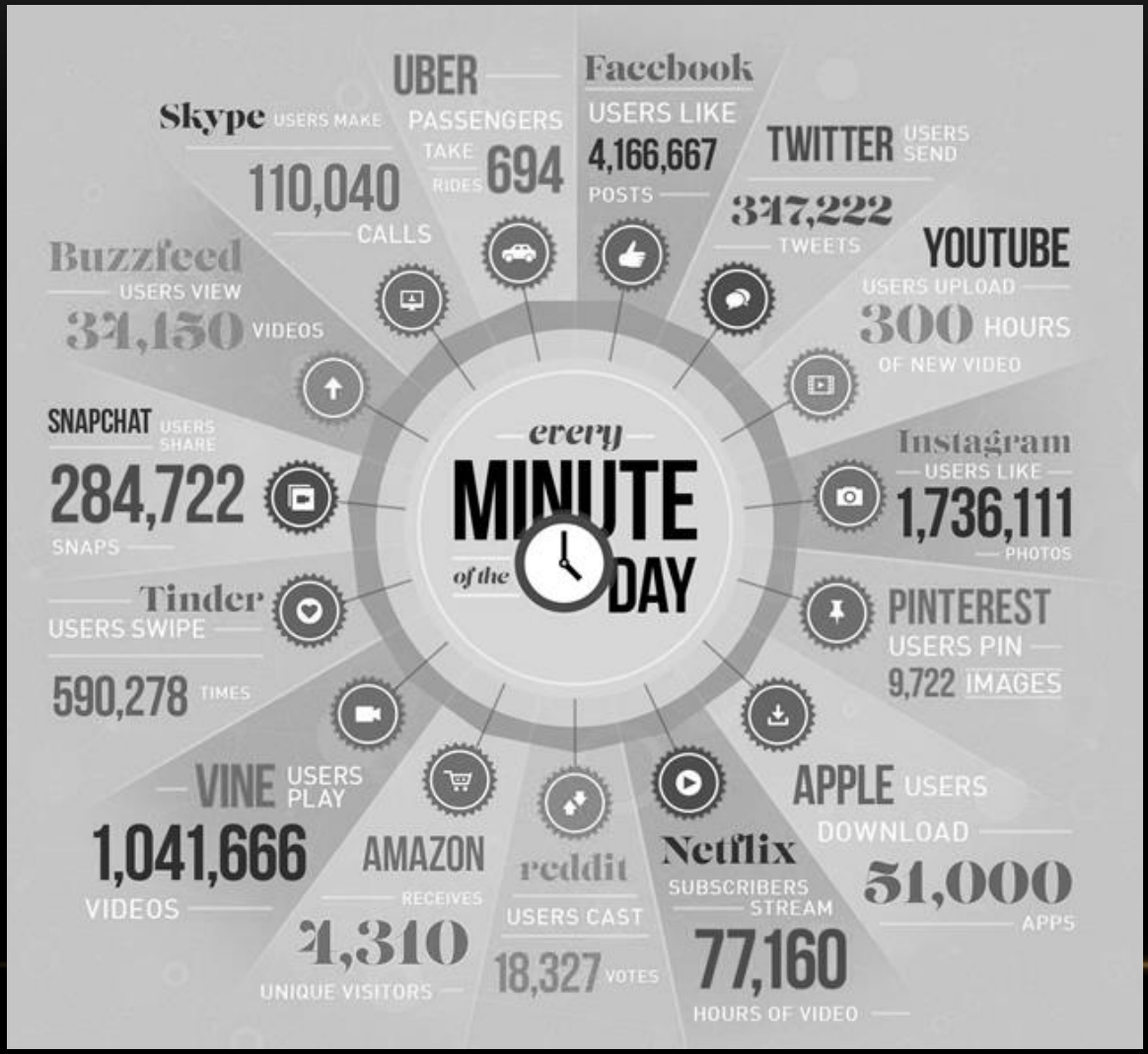
- Introduction to Big Data
  - Concepts and Key Terms, Definition of big data, Few Examples, Use case description
- Challenges of big data
  - Capture/acquire, Storage, Processing, Prediction, Visualization
- Big data tool box
  - Tools for data loading, Tools for data processing, Tools for programming, Tools for visualization
- Hadoop as a big data tool
  - Hadoop basics, Sample program, Benefits and limitations
- Spark as a big data tool
  - Spark basics, Sample program, Benefits and limitations
- Developing Spark API
  - Spark R
  - PySpark

INTRODUCTION TO BIG DATA

- Definition of Big data
  - Big data is data that exceeds the processing capacity of conventional database systems.
  - The data is too big, moves too fast, or doesn't fit the strictures of your database architectures.
  - To gain value from this data, you must choose an alternative way to process it.

INTRODUCTION TO BIG DATA

- Volume -> Size -> Quantity of Data



INTRODUCTION TO BIG DATA

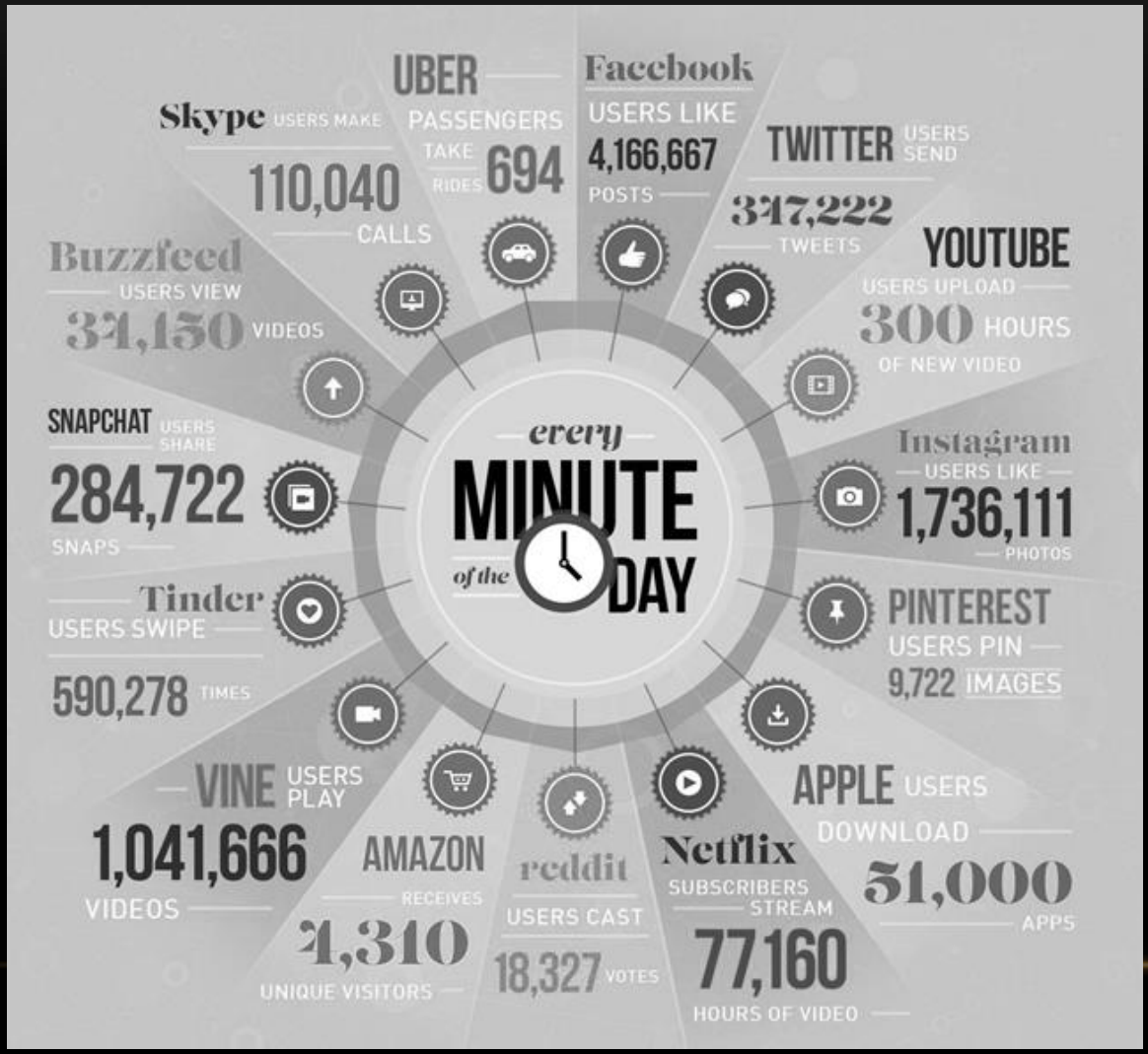
- Velocity -> Frequency of update

INTRODUCTION TO BIG DATA

- Variety -> Diversity

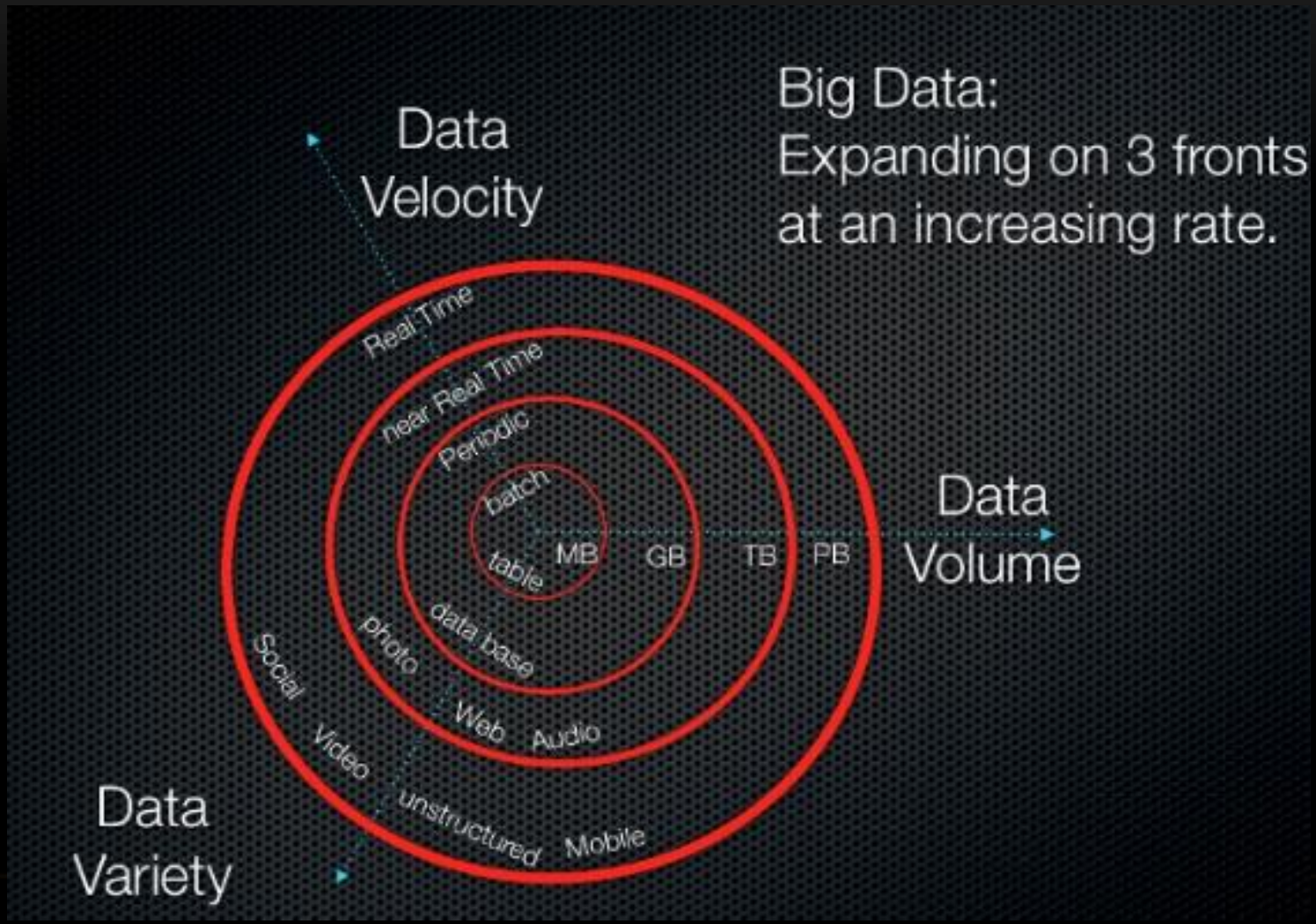
INTRODUCTION TO BIG DATA

- Veracity -> Cleanliness



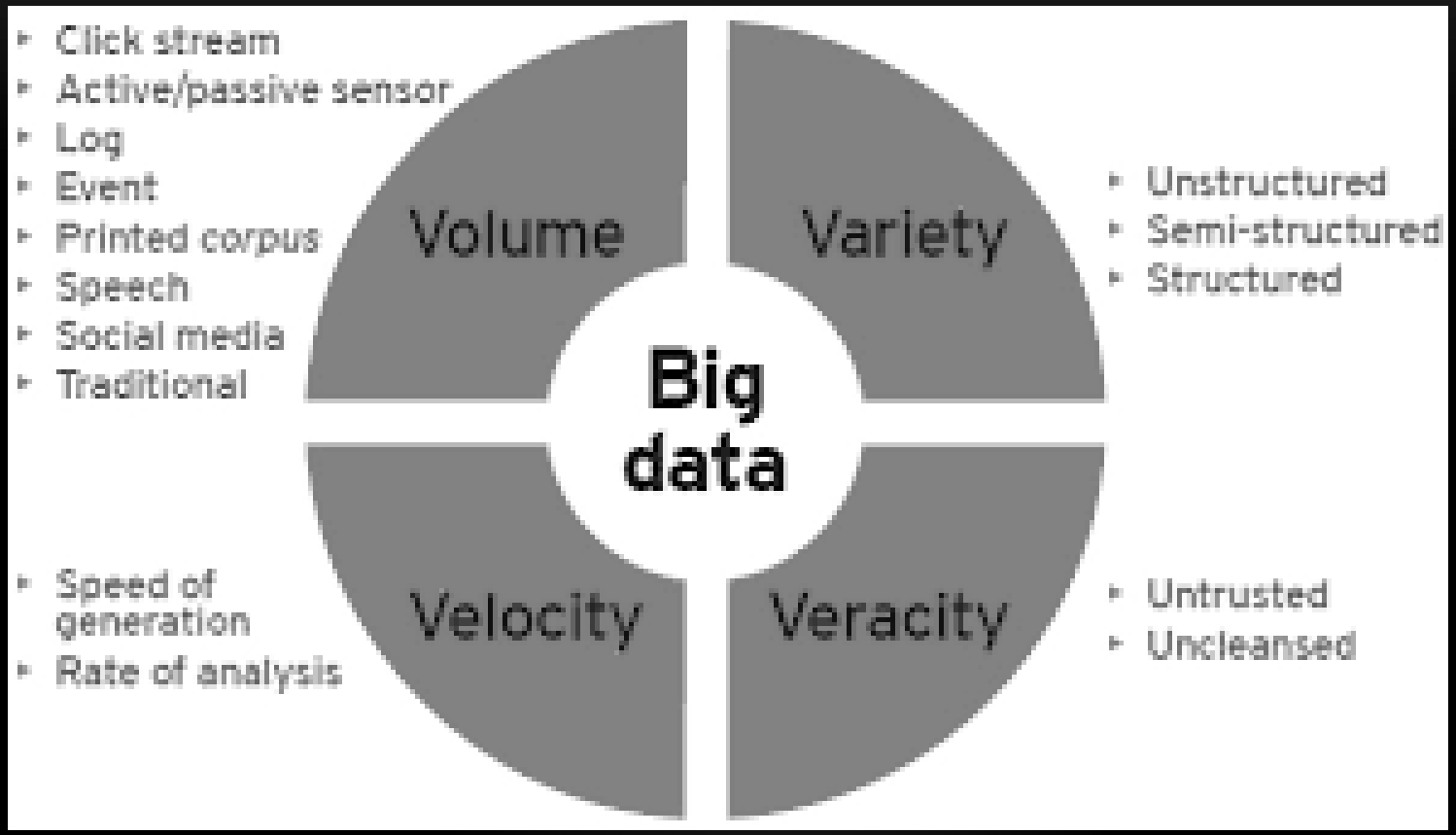
INTRODUCTION TO BIG DATA

- Definition of Big data



INTRODUCTION TO BIG DATA

- 4 V's



INTRODUCTION TO BIG DATA

- Definition of Big data

  - “Big data” is high-volume, -velocity and -variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making

  - By Gartner

KEY TERMINOLOGY IN BIG DATA

- Algorithm:
  - A mathematical formula or statistical process run by software to perform an analysis of data.
  - It usually consists of multiple calculations steps and can be used to automatically process data or solve problems.

KEY TERMINOLOGY IN BIG DATA

- Cloud:
  - Cloud computing, or computing “in the cloud”, simply means software or data running on remote servers, rather than locally.
  - Data stored “in the cloud” is typically accessible over the internet, wherever in the world the owner of that data might be.

KEY TERMINOLOGY IN BIG DATA

- Distributed File System:
  - Data storage system designed to store large volumes of data across multiple storage devices (often cloud based commodity servers), to decrease the cost and complexity of storing large amounts of data.

KEY TERMINOLOGY IN BIG DATA

- Hadoop Distributed File System (HDFS™):
  - HDFS is a distributed file system that provides high-throughput access to application data. Data in a Hadoop cluster is broken down into smaller pieces (called blocks) and distributed throughout the cluster. In this method, the map and reduce functions can be executed on smaller subsets of your larger data sets, and this provides the scalability needed for Big Data processing.

KEY TERMINOLOGY IN BIG DATA

- Data Scientist:
  - Term used to describe an expert in extracting insights and value from data.
  - It is usually someone that has skills in analytics, computer science, mathematics, statistics, creativity, data visualization and communication as well as business and strategy.

KEY TERMINOLOGY IN BIG DATA

- Structured vs. Unstructured Data:
  - Structured data is basically anything than can be put into a table and organized in such a way that it relates to other data in the same table.
  - Unstructured data is everything that can't – email messages, social media posts and recorded human speech

KEY TERMINOLOGY IN BIG DATA

- NoSQL:
  - Refers to database management systems that do not (or not only) use relational tables generally used in traditional database systems.
  - It refers to data storage and retrieval systems that are designed for handling large volumes of data but without tabular categorization (or schemas).

EXAMPLES IN BIG DATA

- Use cases:
  - Banks while issuing credit cards have the luxury to analyze loads of information to reduce the chance of credit card fraud.
  - Cricket websites like ESPN and Cricbuzz have been able to predict how the bowler will bowl or what kind of shot a batsman will play.
  - Retail and E-commerce industries have used Big Data for the prediction of product demands, for analyzing consumer behavior patterns etc.

CHALLENGES OF BIG DATA

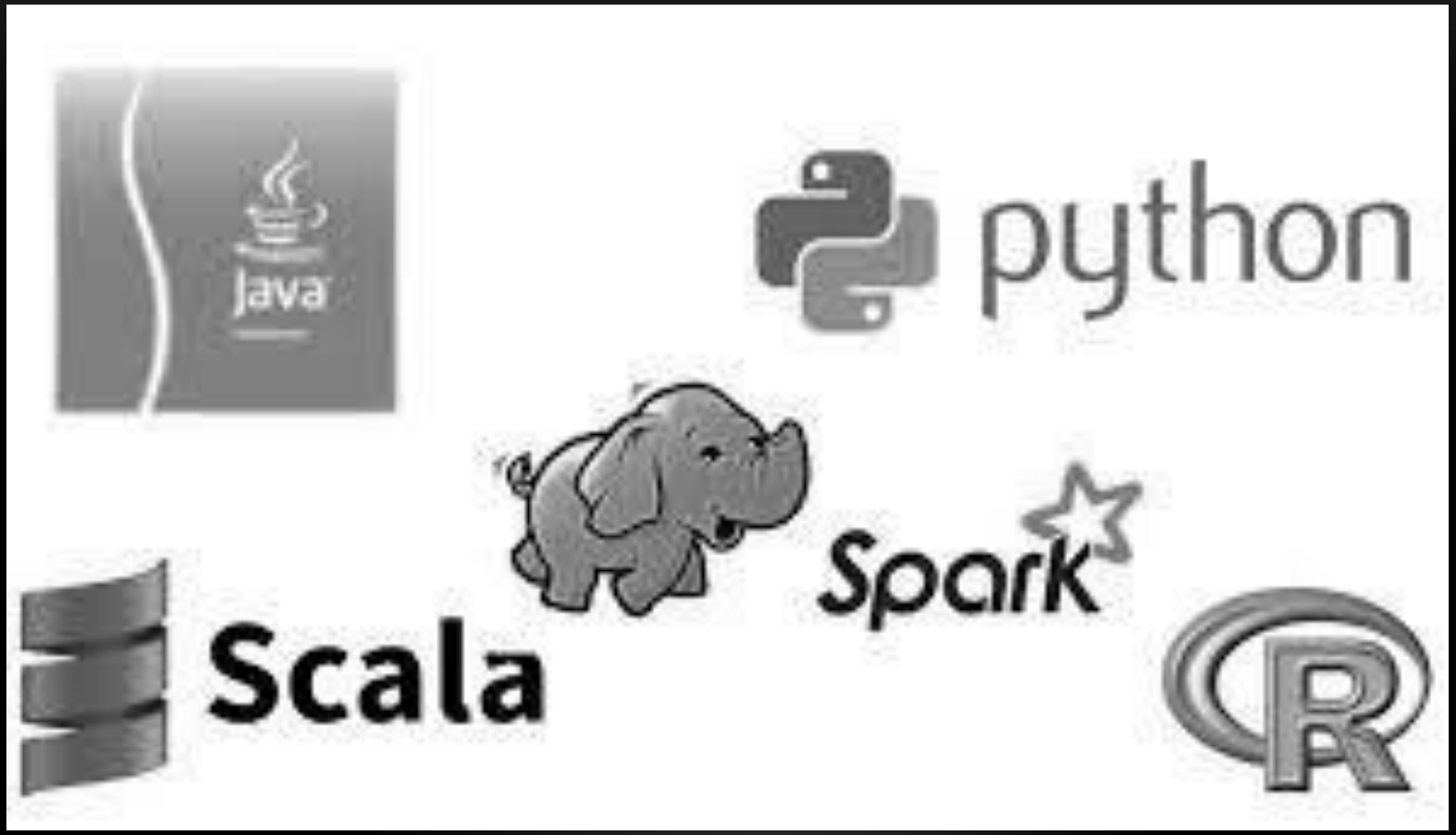
- The major challenges associated with big data are as follows:
  - Capturing data
  - Curation
  - Storage
  - Searching
  - Sharing
  - Transfer
  - Analysis
  - Presentation

CHALLENGES OF BIG DATA

- The major challenges associated with big data are as follows:
  - Identification of a suitable storage for Big Data
  - Data ingestion
  - Data cleaning and processing (Exploratory data analysis)
  - Visualization of the data
  - Apply the machine learning algorithms (If required)

PROGRAMMING IN BIG DATA

- Language in Big data



TOOLS TO TAME BIG DATA

- Hadoop vs. Spark

HADOOP IN BIG DATA

HADOOP IN BIG DATA

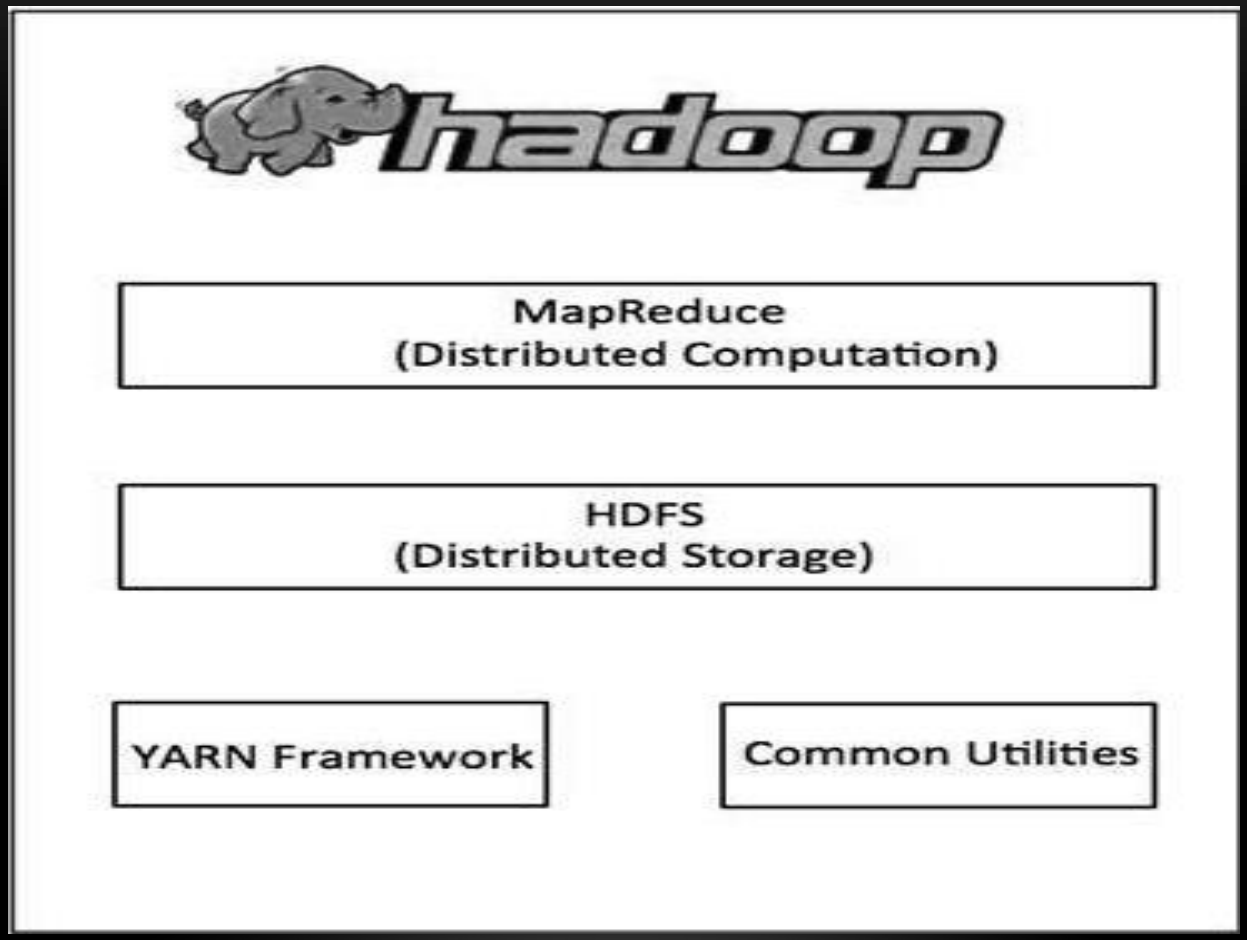
- What is Hadoop?
  - Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models.
  - It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

HADOOP IN BIG DATA

- What is Hadoop?
  - Hadoop is a cluster computing framework developed to solve the problem of Big Data. It is an open-source Java based platform for handling Big Data. It is a combination of two components, HDFS and MapReduce. HDFS is for storing the data while MapReduce is for processing the data.

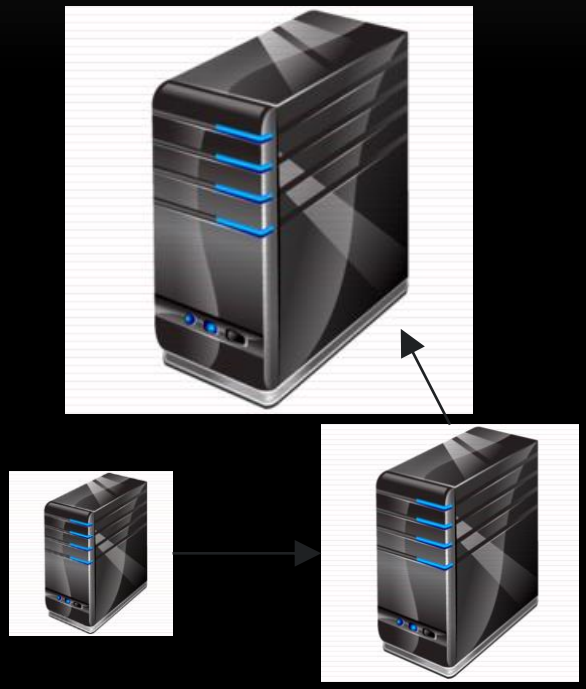
HADOOP IN BIG DATA

- What is Hadoop?



SCALABILITY OF BIG DATA

- Scale Up vs. Scale Out



HADOOP IN BIG DATA

- What is Map Reduce?
  - Hadoop MapReduce is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.
  - The main advantage of the MapReduce framework is its fault tolerance, where periodic reports from each node in the cluster are expected as soon as the work is completed.

HADOOP IN BIG DATA

- The Map Task:
  - This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
- The Reduce Task:
  - This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples.
  - The reduce task is always performed after the map task.

HADOOP IN BIG DATA

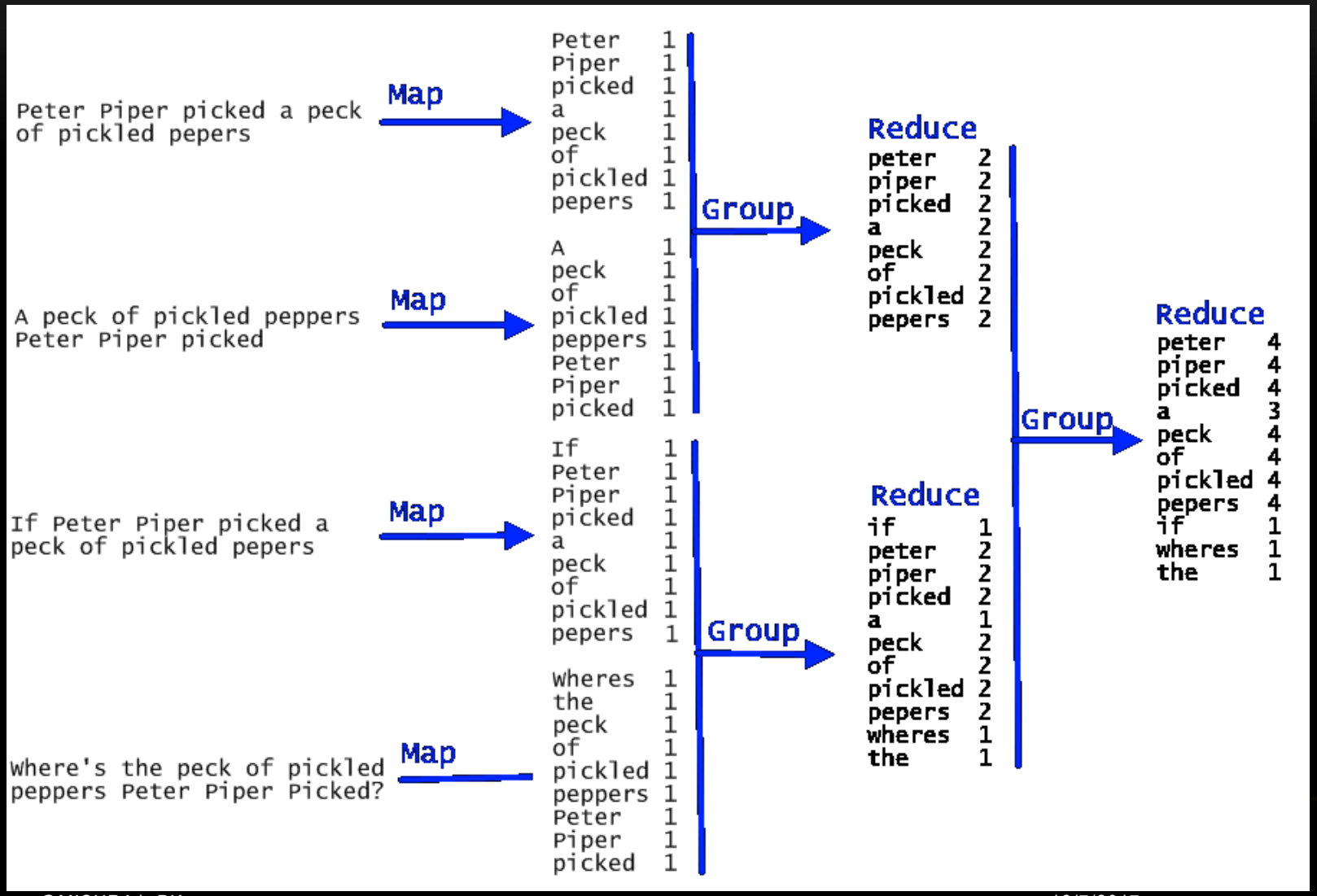
- Pig:
  - Pig is a data flow language that allows users to write complex MapReduce operations in simple scripting language. Then Pig transforms those scripts into a MapReduce job.
- Hive:
  - Apache Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism for querying the data using a SQL-like language called HiveQL. At the same time, this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

HADOOP IN BIG DATA

- Scoop:
  - Enterprises that use Hadoop often find it necessary to transfer some of their data from traditional relational database management systems (RDBMSs) to the Hadoop ecosystem.
  - Sqoop, an integral part of Hadoop, can perform this transfer in an automatic fashion
- Flume:
  - Flume is a service for streaming logs into Hadoop. Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS).

HADOOP IN BIG DATA

- How does MapReduce work:



HADOOP IN BIG DATA

- How does MapReduce work:
- #Setting up RHadoop
- Sys.setenv(HADOOP_CMD="/usr/lib/hadoop-2.2.0/bin/hadoop")
- Sys.getenv("HADOOP_CMD")

- Sys.setenv(HADOOP_STREAMING="/usr/lib/hadoop-2.2.0/share/hadoop/tools/lib/hadoop-streaming-2.2.0.jar")
- Sys.getenv("HADOOP_STREAMING")

- Sys.setenv(JAVA_HOME="/usr/lib/jvm/jdk1.7.0_67")
- Sys.getenv("JAVA_HOME")

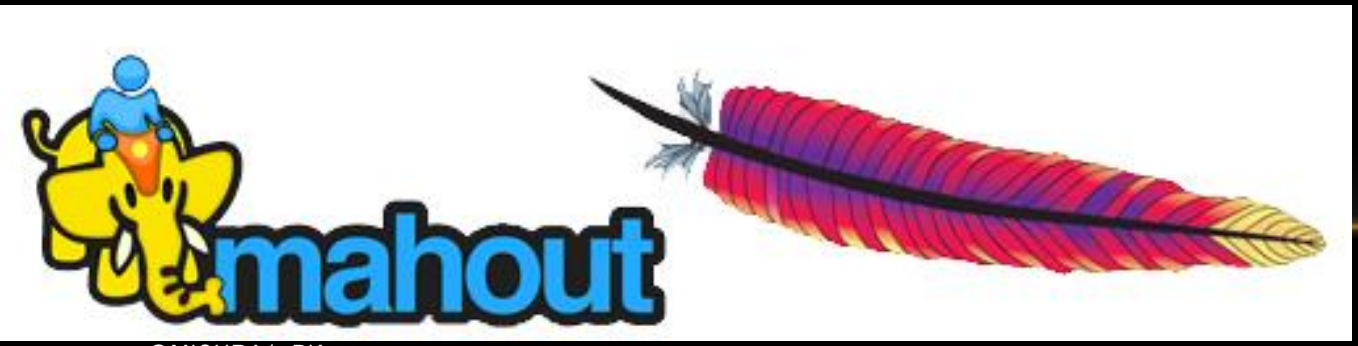
- install.packages(c("codetools", "R", "Rcpp", "RJSONIO", "bitops"))
- install.packages(c("digest", "functional", "stringr", "plyr", "reshape2", "rJava"))
- install.packages("rmr2")
- install.packages(("caTools"))
- install.packages(("rJava"))

HADOOP IN BIG DATA

- How does MapReduce work:
- `integers<-1:1000`
- `data<-to.dfs(small.integers,"small_ints3")`
- `square_func<-function(k,v){v^2}`
- `mapreduce(input="/small_ints3",map=square_func,output="/data/op3")`
- `results<-from.dfs("/data/op3")`
- `results`

APACHE MAHOUT FOR BIG DATA

- Apache Mahout:
  - It is an open source project that is primarily used for creating scalable machine learning algorithms.
  - It implements popular machine learning techniques such as:
    - Recommendation
    - Classification
    - Clustering



APACHE MAHOUT FOR BIG DATA

- Challenges:
  - Most of the libraries deprecated

Classification *with CLI drivers*			
Logistic Regression - trained via SGD	*deprecated*		
Naive Bayes / Complementary Naive Bayes		*deprecated*	x
Hidden Markov Models	*deprecated*		
Clustering *with CLI drivers*			
Canopy Clustering	*deprecated*	*deprecated*	
k-Means Clustering	*deprecated*	*deprecated*	
Fuzzy k-Means	*deprecated*	*deprecated*	
Streaming k-Means	*deprecated*	*deprecated*	
Spectral Clustering		*deprecated*	

SPARK IN BIG DATA

SPARK FOR BIG DATA

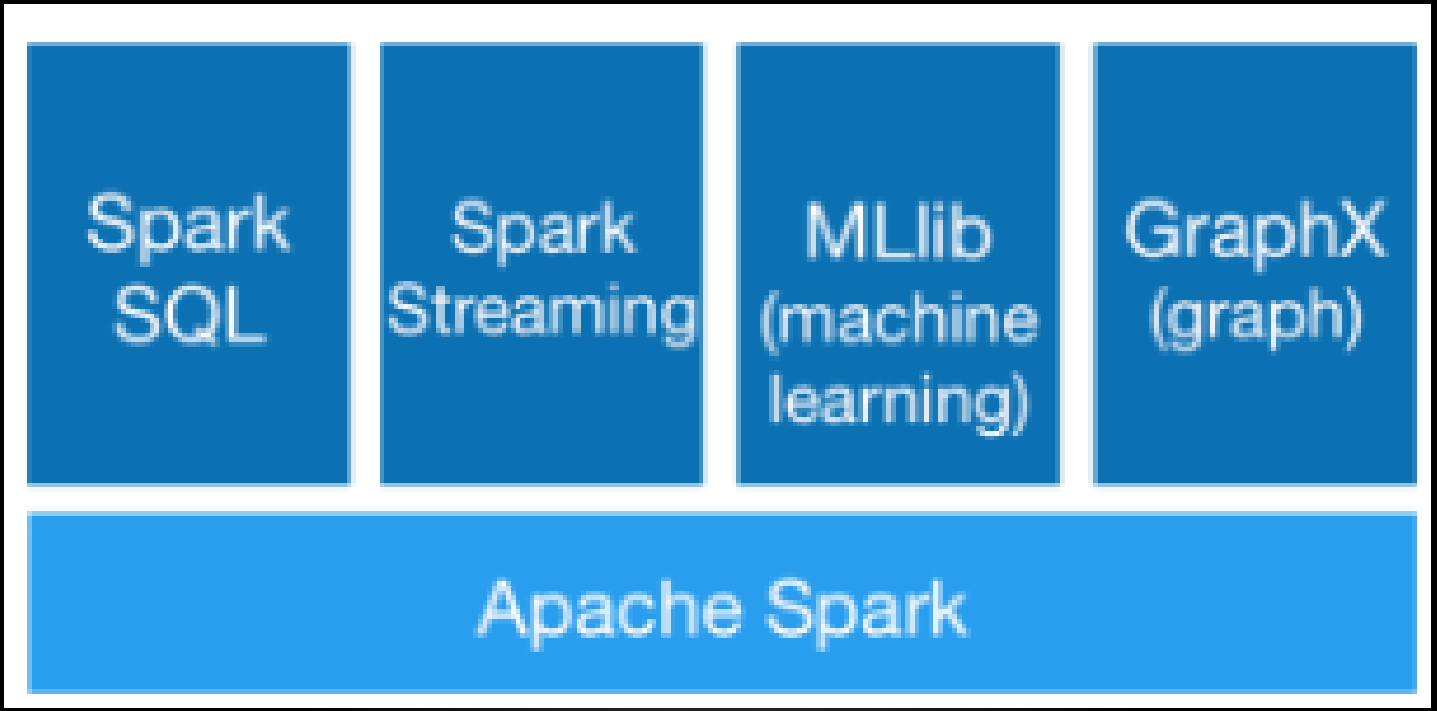
- Spark for Big data:
  - Apache Spark is a lightning-fast cluster computing designed for fast computation.

SPARK FOR BIG DATA

- Spark for Big data:
  - It was built on top of Hadoop MapReduce and it extends the MapReduce model to efficiently use more types of computations which includes Interactive Queries and Stream Processing.

SPARK FOR BIG DATA

- Spark Core:
  - Spark was introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process.



SPARK FOR BIG DATA

- Spark for Big data:
  - Spark is not a modified version of Hadoop and is not, really, dependent on Hadoop because it has its own cluster management.
  - Hadoop is just one of the ways to implement Spark.

SPARK FOR BIG DATA

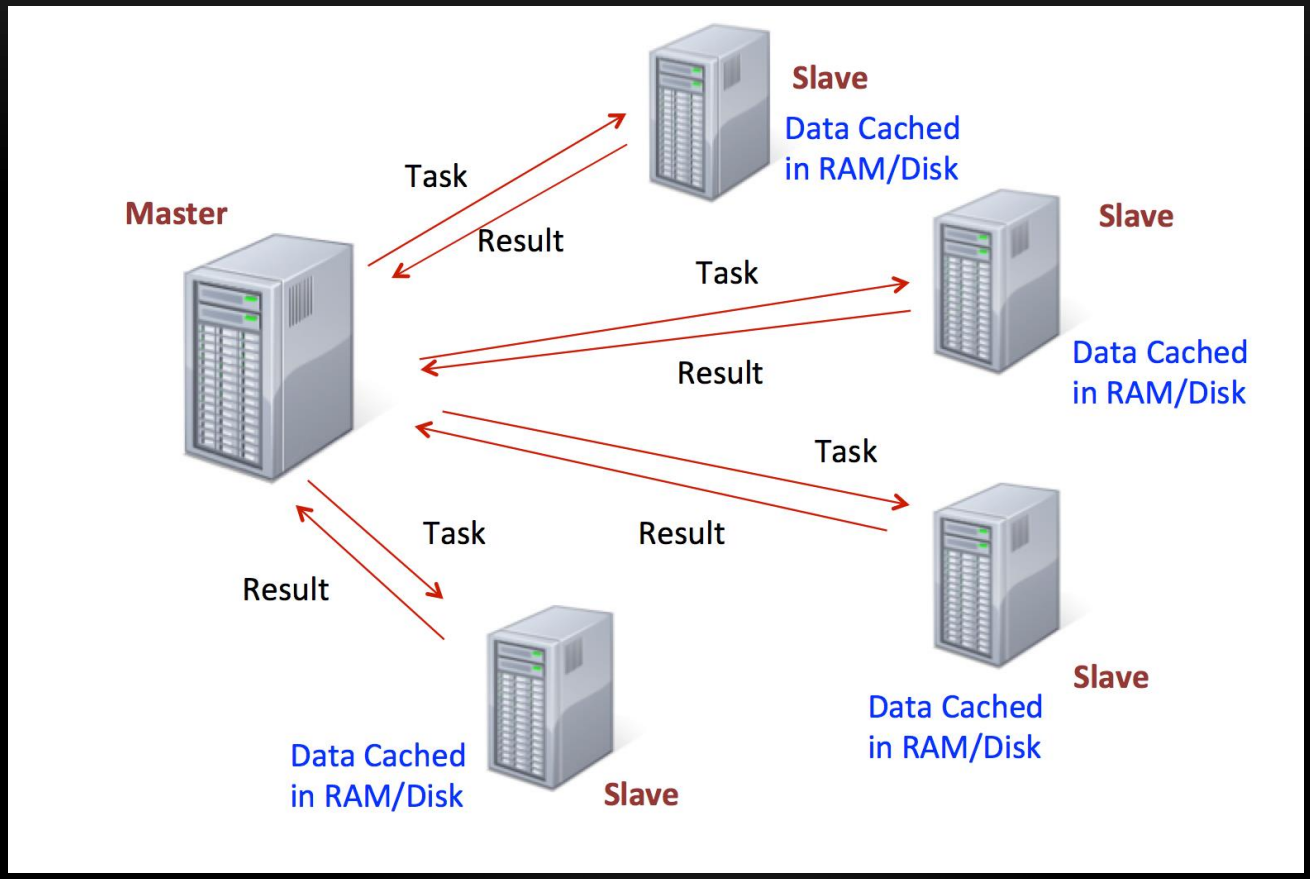
- Spark for Big data:
  - The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

SPARK FOR BIG DATA

- Spark for Big data:
  - Spark uses Hadoop in two ways – one is storage and second is processing.
  - Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only.

SPARK FOR BIG DATA

- How does Spark work:



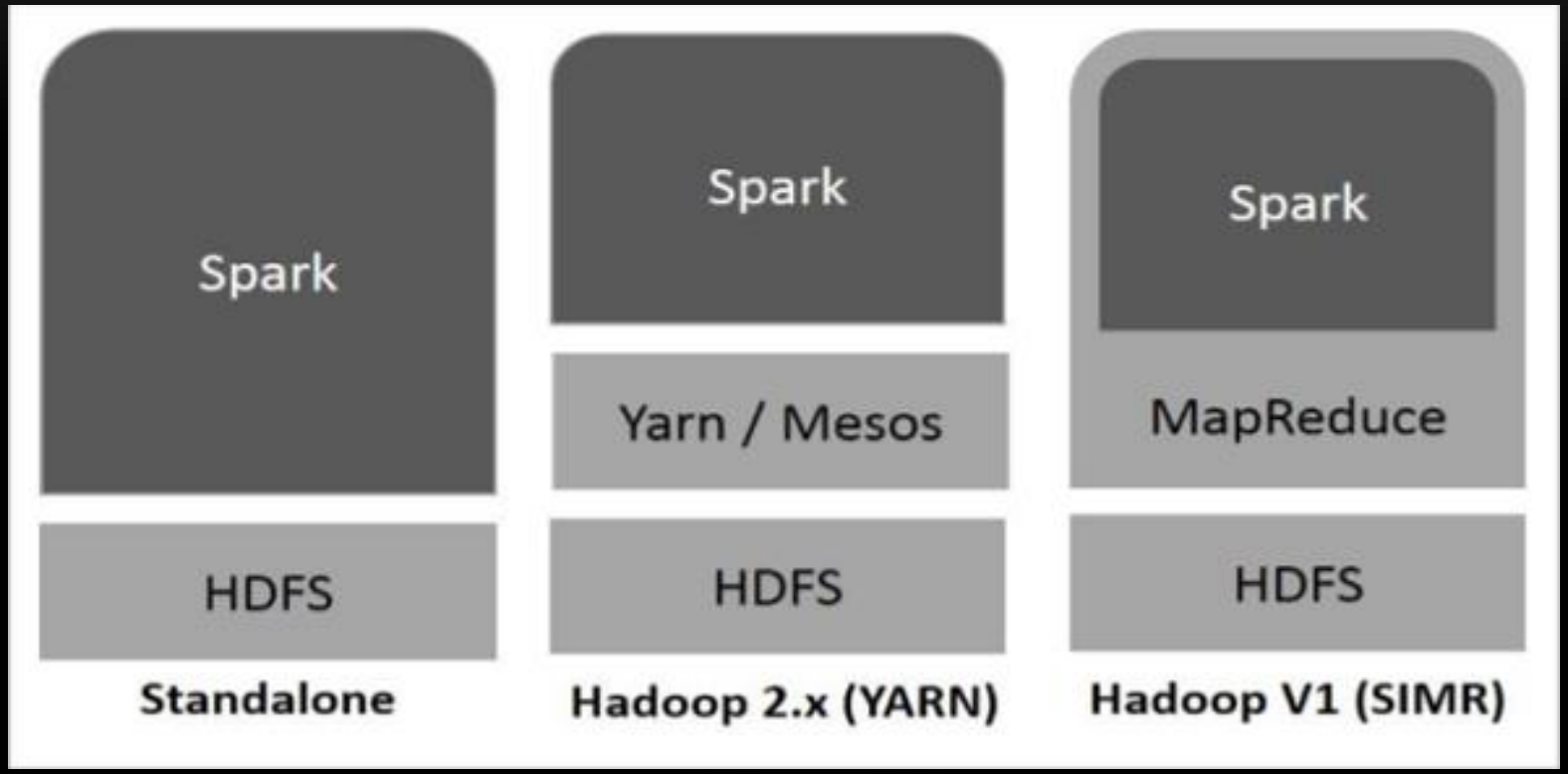
SPARK FOR BIG DATA

- Features of Spark:
  - Speed − Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk.
  - This is possible by reducing number of read/write operations to disk.
  - It stores the intermediate processing data in memory.

SPARK FOR BIG DATA

- Features of Spark:
  - Supports multiple languages − Spark provides built-in APIs in Java, Scala, R, or Python, to write applications in different languages.
  - Spark comes up with 80 high-level operators for interactive querying.
  - Advanced Analytics − Spark not only supports ‘Map’ and ‘reduce’.
  - It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

SPARK DEPLOYMENT FOR BIG DATA



SPARK DEPLOYMENT FOR BIG DATA

- Spark Deployment in three ways:
- 1- Standalone −
  - Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly.
  - Spark and MapReduce will run side by side to cover all spark jobs on cluster.

SPARK DEPLOYMENT FOR BIG DATA

- Spark Deployment in three ways:
- 2- Hadoop Yarn −
  - Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required.
  - It helps to integrate Spark into Hadoop ecosystem or Hadoop stack.
  - It allows other components to run on top of stack.

SPARK DEPLOYMENT FOR BIG DATA

- Spark Deployment in three ways :
- 3- Spark in MapReduce (SIMR) −
  - Spark in MapReduce is used to launch spark job in addition to standalone deployment.
  - With SIMR, user can start Spark and uses its shell without any administrative access.

COMPONENTS OF SPARK

- Spark SQL:
  - Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

COMPONENTS OF SPARK

- Spark Streaming:
  - Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics.
  - It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

COMPONENTS OF SPARK

- MLlib (Machine Learning Library):
  - MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture.
  - It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations.
  - Spark MLlib is nine times as fast as the Hadoop disk-based version of **Apache Mahout** (before Mahout gained a Spark interface).

COMPONENTS OF SPARK

- GraphX:
  - GraphX is a distributed graph-processing framework on top of Spark.
  - It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API.
  - It also provides an optimized runtime for this abstraction.

UNDERSTANDING SPARK

RDD DESCRIPTION

- What is RDD?
  - Resilient because RDDs are immutable(can't be modified once created)
  - Distributed because it is distributed across cluster
  - Dataset because it holds data.
  - Each RDD is split into multiple partitions, which may be computed on different nodes of the cluster.
  - The Spark RDDs can contain any type of Python, Java or Scala objects, including user-defined classes.
  - They are not actual data, but they are Objects, which contains information about data residing on the cluster.

RDD DESCRIPTION

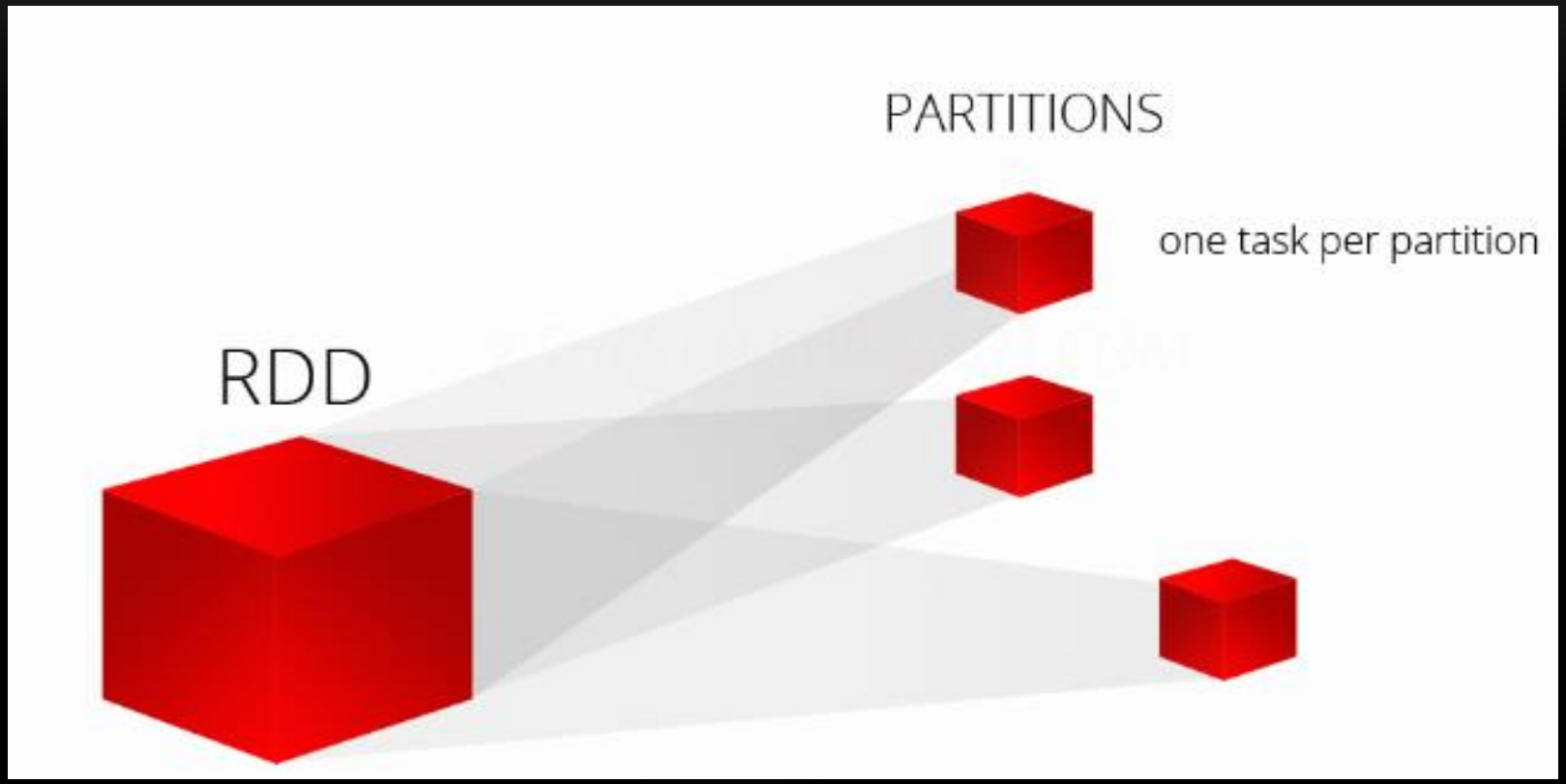
- Features of RDD:
  - In Hadoop, we store the data as blocks and store them in different data nodes. In Spark, we make partitions of the RDDs and store in worker nodes (data nodes) which are computed in parallel across all the nodes.
  - In Hadoop, we need to replicate the data for fault recovery, but in case of Spark, replication is not required as this is performed by the RDDs.
  - RDDs load the data for us and are resilient, which means they can be recomputed.
  - RDDs perform two types of operations: Transformations, which creates a new dataset from the previous RDD and Actions, which return a value to the driver program after performing the computation on the dataset.
  - RDDs keeps track of Transformations and checks them periodically. If a node fails, it can rebuild the lost RDD partition on the other nodes, in parallel

RDD DESCRIPTION

- What is RDD?

RDD DESCRIPTION

- How it works?



RDD DESCRIPTION

- How it works?
  - RDDs are divided into smaller chunks called Partitions, and when you execute some action, a task is launched per partition.
  - So it means, the more the number of partitions, the more the parallelism.

RDD DESCRIPTION

- How it works?
  - Spark automatically decides the number of partitions that an RDD has to be divided into but you can also specify the number of partitions when creating an RDD.
  - These partitions of an RDD is distributed across all the nodes in the network.

SPARK DEMO - DATA MANAGEMENT

DATA SHARING USING SPARK RDD

- Why RDD is best?
  - Data sharing is slow in MapReduce due to replication, serialization, and disk IO.
  - Most of the Hadoop applications, they spend more than 90% of the time doing HDFS read-write operations.
  - Recognizing this problem, researchers developed a specialized framework called Apache Spark.

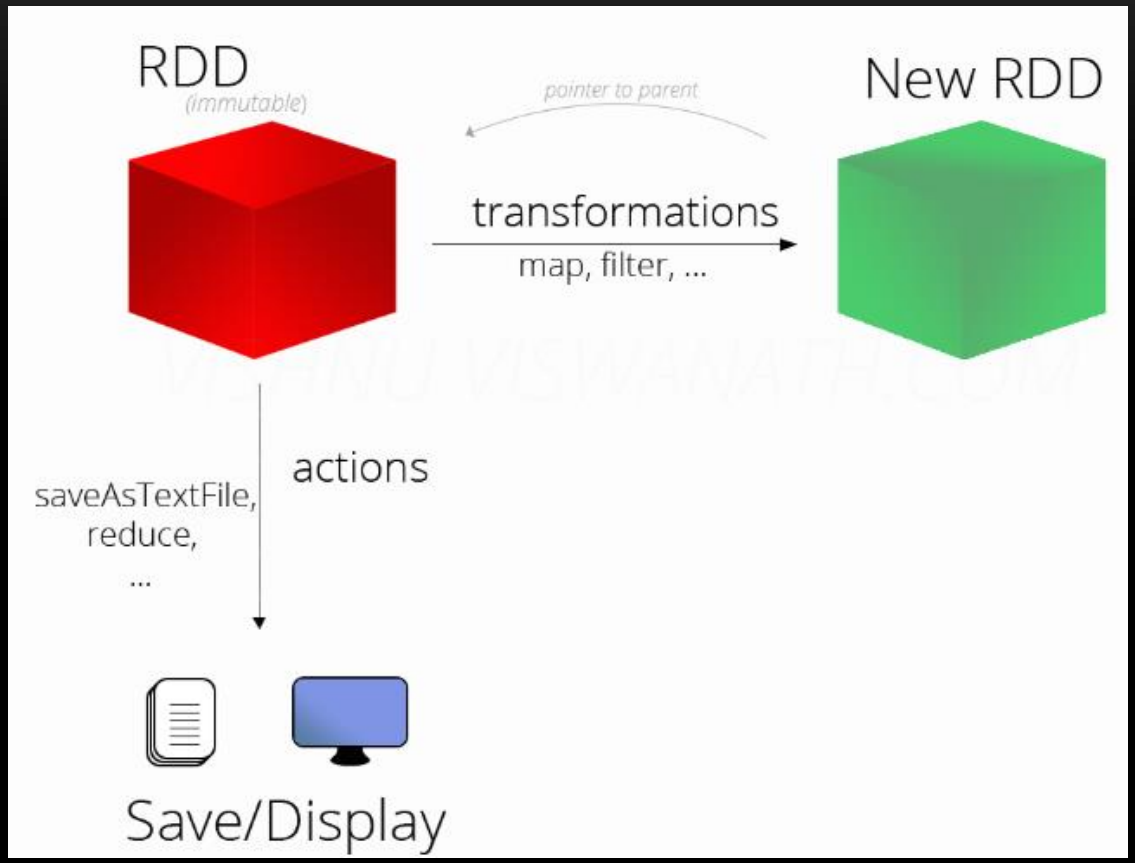
DATA SHARING USING SPARK RDD

- Why RDD is best?
  - The key idea of spark is Resilient Distributed Datasets (RDD);
  - it supports in-memory processing computation.
  - It stores the state of memory as an object across the jobs and the object is sharable between those jobs.
  - Data sharing in memory is 10 to 100 times faster than network and Disk.

SAMPLE SPARK TRANSFORMATIONS

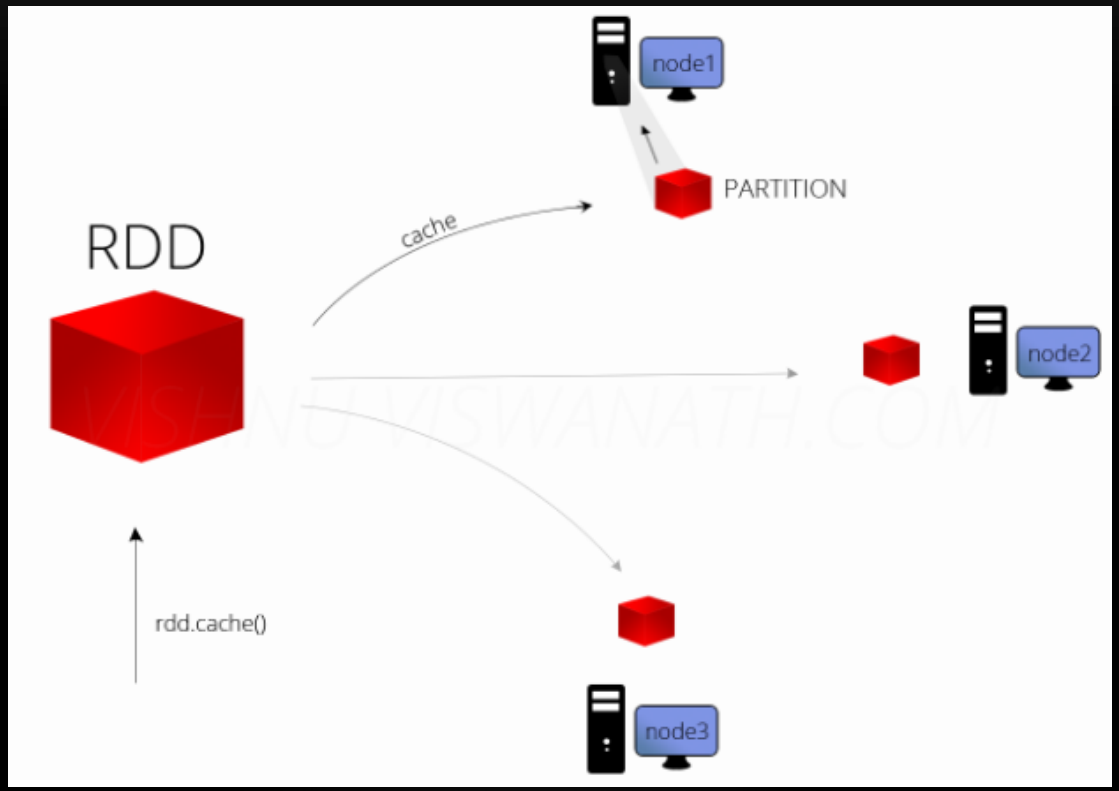
- map(func):
  - Return a new distributed dataset formed by passing each element of the source through a function func.
- filter(func):
  - Return a new dataset formed by selecting those elements of the source on which func returns true
- union(otherDataset):
  - Return a new dataset that contains the union of the elements in the source dataset and the argument.

SAMPLE SPARK TRANSFORMATIONS



SAMPLE SPARK TRANSFORMATIONS

- Caching



SAMPLE SPARK TRANSFORMATIONS

- Caching
  - You can cache an RDD in memory by calling rdd.cache().
  - When you cache an RDD, it's Partitions are loaded into memory of the nodes that hold it.
  - Caching improves the performance of your application
  - Caching stores the computed result of the RDD in the memory thereby eliminating the need to recompute it every time.

SPARK VS. HADOOP

- Spark performs in-memory operations by copying the data from distributed storage into RAM memory which is much faster. As a result of this, the time consumed to read and write is reduced.
- With in-memory caching abstraction, Spark caches input data sets in-memory and as a result, each operation does not warrant the data to be read from disk

SPARK VS. HADOOP

- Spark doesn't have its own distributed file system, but can use HDFS as its underlying storage.
- Spark has its own machine learning libraries, called MLib, whereas Hadoop system must be interfaced with a third-party machine learning library, for example, Apache Mahout.

SPARK VS. HADOOP

- If the data operation and reporting requirements are static, MapReduce will perform batch-mode processing to process the same data. But if analytics has to be performed on streaming data, like sensor data or data from applications requiring multiple operations, then Spark would be an apt choice to process the same.

SPARK VS. HADOOP

- Spark's functionality for handling advanced data processing tasks such as real time stream processing and machine learning is far more advanced than Hadoop alone. This, along with the speed provided by in-memory operations and high efficiency in handling real time data are some of the reasons for its popularity.

MACHINE LEARNING- USING SPARK

NEXT