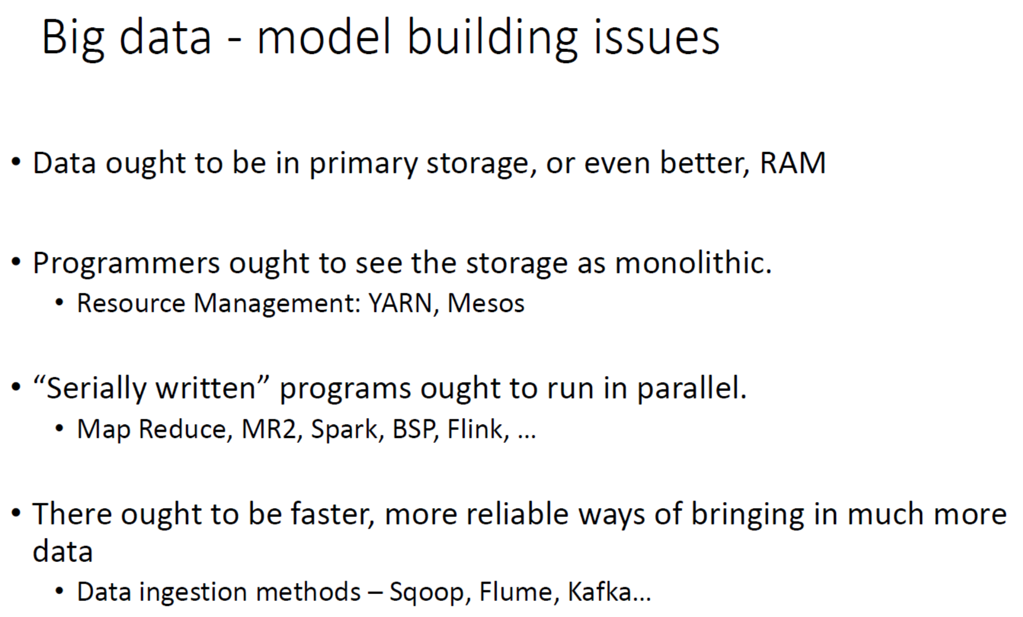
BIG DATA

What is **Big Data** in simple words?
= The answer is in V's - Velocity(speed), Variety (unstructured and structured), Veracity(untrust, unclean) and volume
Any data which has this V in it, is called big data.

Why do we need big data in data science?
Answer is simple: the more data the better are the insights, more power decisions.

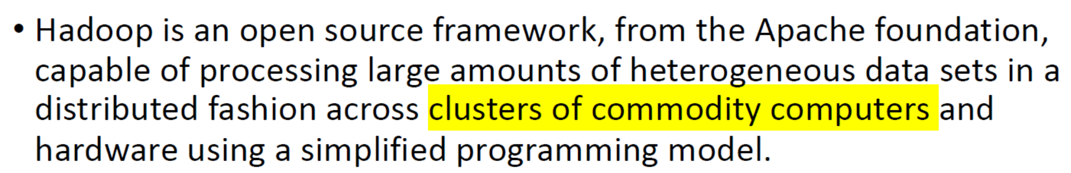


Big data - model building issues

- Data ought to be in primary storage, or even better, RAM
- Programmers ought to see the storage as monolithic.
  - Resource Management: YARN, Mesos
- "Serially written" programs ought to run in parallel.
  - Map Reduce, MR2, Spark, BSP, Flink, …
- There ought to be faster, more reliable ways of bringing in much more data
  - Data ingestion methods – Sqoop, Flume, Kafka…

Few terms you should know about the operation done in backend:
To scale horizontal means to add a node.
To scale vertically means to add resources to a single node.

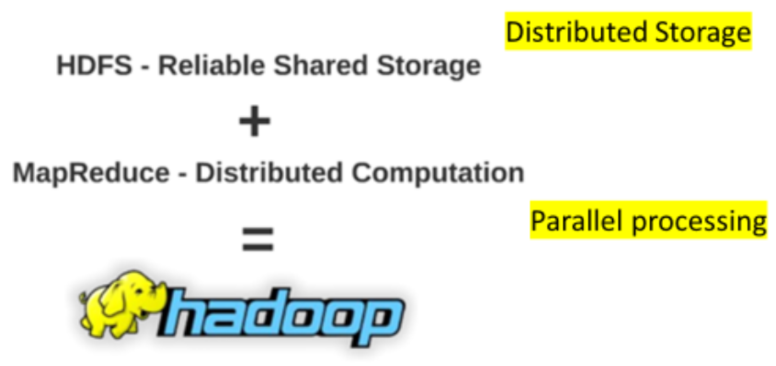
What is Hadoop?



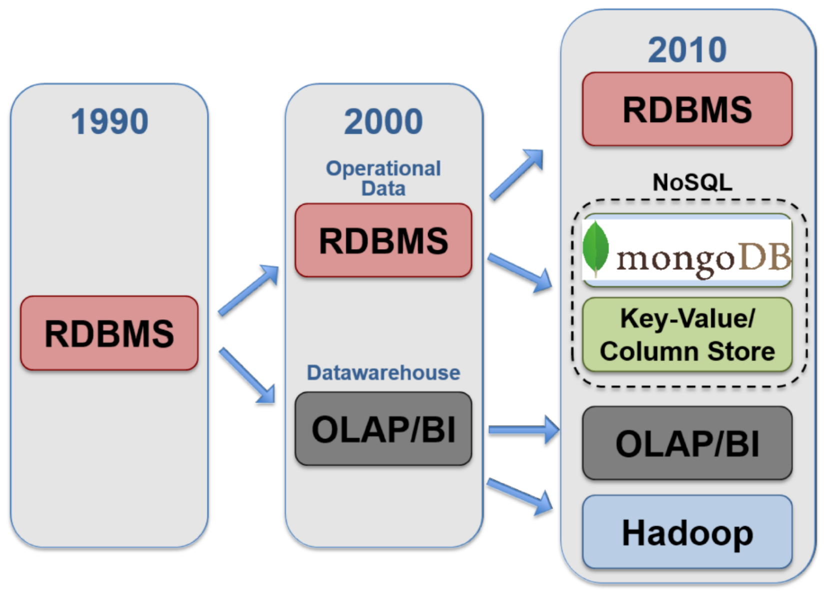
- Hadoop is an open source framework, from the Apache foundation, capable of processing large amounts of heterogeneous data sets in a distributed fashion across clusters of commodity computers and hardware using a simplified programming model.

In short,
Hadoop is an open source framework capable of parallel processing with distributed storage.

HDFS and MapReduce


Distributed Storage
HDFS - Reliable Shared Storage
+
MapReduce - Distributed Computation
Parallel processing
=
hadoop

Things to be known



Transition from databases to data warehouses to data lakes

40

The new Big Data thinking

- All data has potential value
- Data hoarding
- No defined schema—stored in native format
- Schema is imposed and transformations are done at query time *(schema-on-read).*
- Apps and users interpret the data as they see fit

BIG DATA

Why do we need Hadoop?

- Scalability: Need to process Multi Petabyte Datasets, quickly.
- Data may not have strict schema
- Expensive to build reliability in each application
- Fault tolerant: Nodes fails everyday
- Need common infrastructure
- Very Large Distributed File System
- Low cost: Assumes Commodity Hardware
- Optimized for Batch Processing
- Runs on heterogeneous OS

Key Hadoop vendors:

- Pure play Hadoop vendors: Cloudera, Hortonworks, MapR, IBM OpenPlatform, Huawei FusionInsight, Seabox, Transwarp
- Cloud infrastructure as a service (IaaS): Hadoop on AWS, Hadoop on Azure
- Platform as a service (PaaS): IBM BigInsights, Microsoft HDInsight, Google Cloud Platform, Amazon EMR, Oracle Big Data Cloud Service, Qbole
- Big Data Appliances: Teradata, Oracle Big Data Appliance, Cray

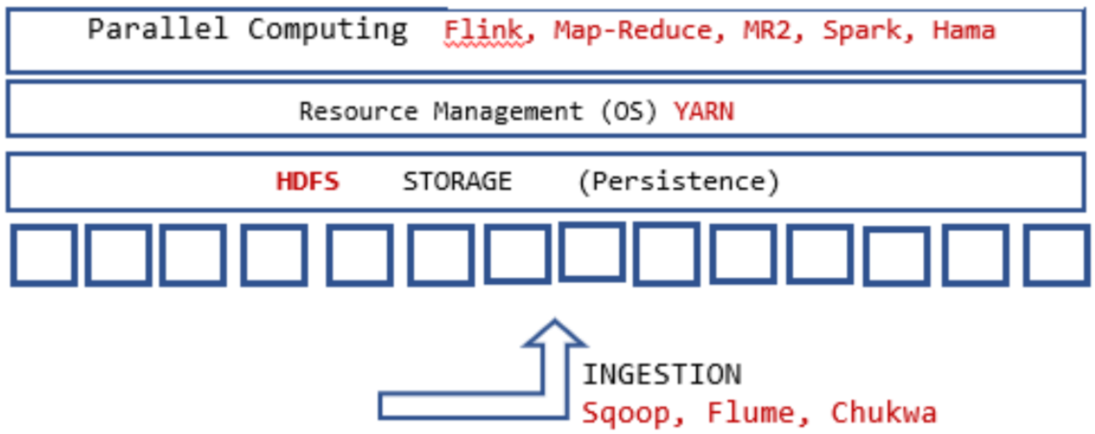
Important components of hadoop:

- HDFS - Hadoop Distributed File System
  - Horizontal scalability. Thousands of servers holding petabytes of data.
  - Commodity hardware. HDFS is designed with relatively cheap commodity hardware in mind. HDFS is self-healing and replicating.
  - Fault tolerance. Every component of the Hadoop ecosystem knows how to deal with hardware failures.
- MapReduce
  - MapReduce takes care of distributed computing. It reads the data, usually from its storage, the Hadoop Distributed File System (HDFS), in an optimal way. However, it can read the data from other places too, including mounted local file systems, the web, and databases. It divides the computations between different computers (servers, or nodes). It is also fault-tolerant.
- HBase, the database for Big Data
  - HBase is a database for Big Data, up to millions of columns and billions of rows.
  - It is a key-value database, not a relational database. Key-value databases are considered as more fitting for Big Data. Why? Because they don't store nulls! This gives them the appellation of "sparse."
- ZooKeeper
  - ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.
  - ZooKeeper is also fault-tolerant.
- Hive - data warehousing
  - Hive defines a simple SQL-like query language, called QL, that enables users familiar with SQL to query the data.
  - Hive allows you to write custom mappers and reducers to extend the QL capabilities.
- Pig - Big Data manipulation
  - Pig Latin is the language of the stored procedures of Big Data. It allows you to manipulate large volumes of information, analyze them, and create new derivative data sets. Internally it creates a sequence of MapReduce jobs, and thus you can use this simple language to solve pretty sophisticated large-scale problems.

BIG DATA

The process followed:

Everyone needs them.....


Parallel Computing Flink, Map-Reduce, MR2, Spark, Hama
Resource Management (OS) YARN
HDFS STORAGE (Persistence)
INGESTION
Sqoop, Flume, Chukwa

Things to be known:

Machine Learning on Spark-ML, Mahout, Samsara, H20, Hadoop

Streaming & Near Real Time Processing — KAFKA, SAMZA, STORM, TRIDENT, SPARK-STREAMING, FLINK

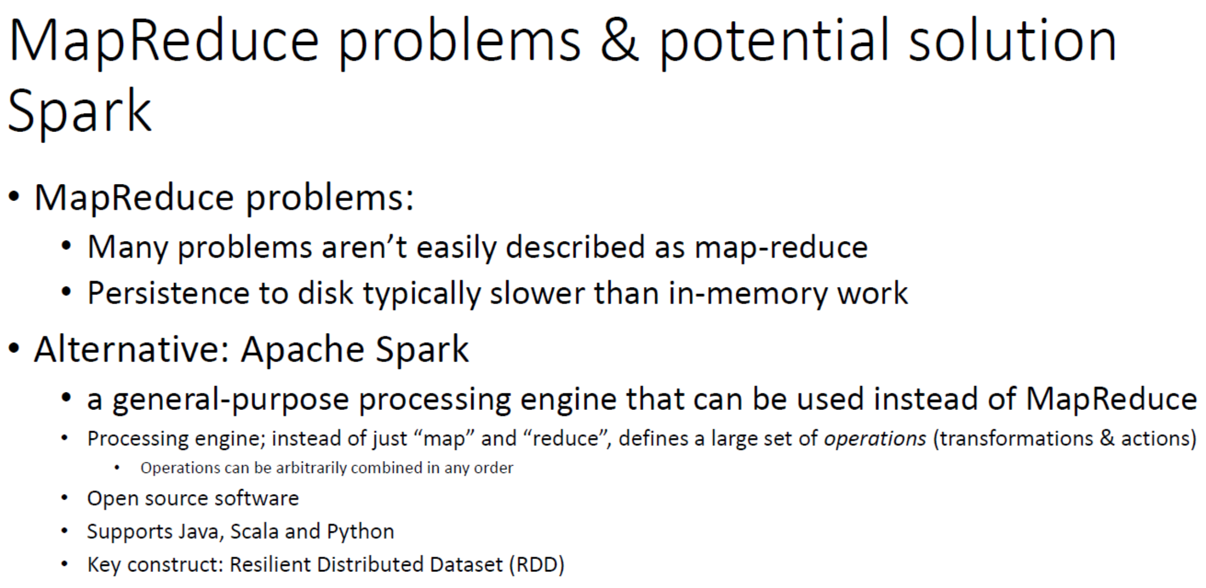
Application Programming — PIG, Oozie, Hadoop Streaming, Spark-R

Data Organization
SQL — HIVE, IMPALA, SPARK SQL, Apache Drill
NoSQL — Hbase, Cassandra, MongoDB, Neo4J, Kudu

Document by - Shubham kotal

SPARK

We know that in Hadoop parallel processing is provided by map reduce.
But, MapReduce has some limitations. To overcome that spark is useful.



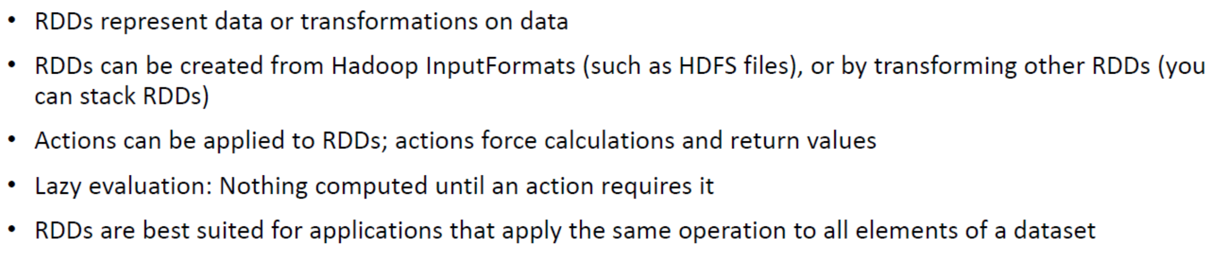
MapReduce problems & potential solution Spark

- MapReduce problems:
  - Many problems aren’t easily described as map-reduce
  - Persistence to disk typically slower than in-memory work
- Alternative: Apache Spark
  - a general-purpose processing engine that can be used instead of MapReduce
  - Processing engine; instead of just “map” and “reduce”, defines a large set of *operations* (transformations & actions)
    - Operations can be arbitrarily combined in any order
  - Open source software
  - Supports Java, Scala and Python
  - Key construct: Resilient Distributed Dataset (RDD)

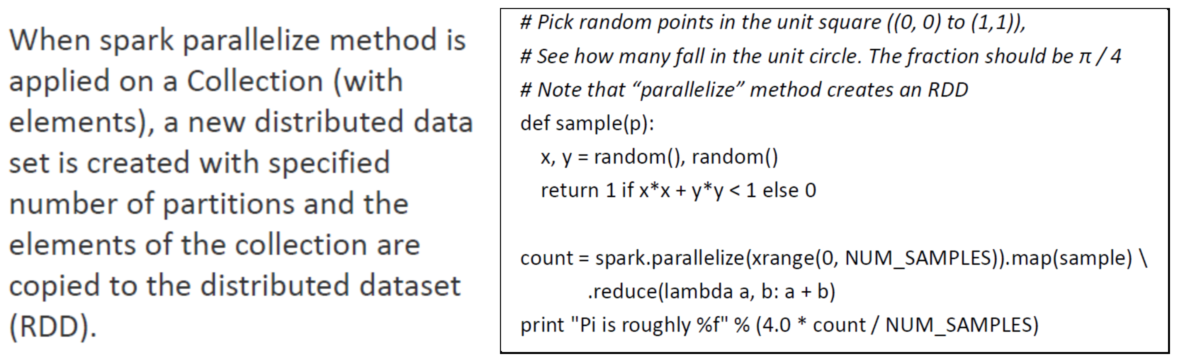
Features of spark

1. Speed
2. Support multiple language
3. Advance analytics (streaming, ml, graph algorithms, sql queries)

RDD: Resilient Distributed Dataset.



- RDDs represent data or transformations on data
- RDDs can be created from Hadoop InputFormats (such as HDFS files), or by transforming other RDDs (you can stack RDDs)
- Actions can be applied to RDDs; actions force calculations and return values
- Lazy evaluation: Nothing computed until an action requires it
- RDDs are best suited for applications that apply the same operation to all elements of a dataset



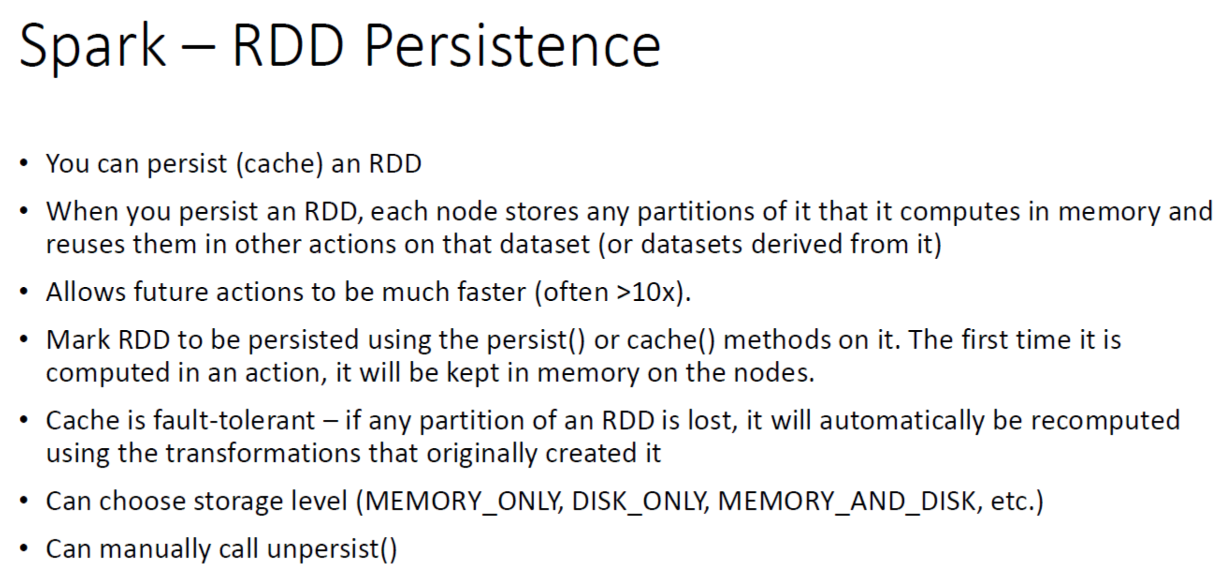
When spark parallelize method is applied on a Collection (with elements), a new distributed data set is created with specified number of partitions and the elements of the collection are copied to the distributed dataset (RDD).

```
# Pick random points in the unit square ((0, 0) to (1,1)),
# See how many fall in the unit circle. The fraction should be π / 4
# Note that “parallelize” method creates an RDD
def sample(p):
   x, y = random(), random()
   return 1 if x*x + y*y < 1 else 0

count = spark.parallelize(xrange(0, NUM_SAMPLES)).map(sample) \
          .reduce(lambda a, b: a + b)
print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

Document by - Shubham kotal

Why we make it RDD: Because of ite persistence



Spark – RDD Persistence

- You can persist (cache) an RDD
- When you persist an RDD, each node stores any partitions of it that it computes in memory and reuses them in other actions on that dataset (or datasets derived from it)
- Allows future actions to be much faster (often >10x).
- Mark RDD to be persisted using the persist() or cache() methods on it. The first time it is computed in an action, it will be kept in memory on the nodes.
- Cache is fault-tolerant – if any partition of an RDD is lost, it will automatically be recomputed using the transformations that originally created it
- Can choose storage level (MEMORY_ONLY, DISK_ONLY, MEMORY_AND_DISK, etc.)
- Can manually call unpersist()

Benefits of spark over map reduce:
Faster
Easier to program
General data preprocessing
Benefits of In-memory computing
Fault recovery

Need not need to know much about hive and pig but just in simple words
Hive - Hive is a data warehouse infrastructure tool to process structured data in Hadoop.
Pig - It's a query language to perform map reduce tasks.

SPARK:

RDD: Whatever done in spark gets saved as RDD, even data frames saved as RDD. As transformation in RDD gets saved, as the name says resilient (tough and flexible) distributed data.

Replication by default is 3. (Immutable, fault tolerant)

YARN in spark behaves as a resource manager which allocates clusters.

Spark is a parallel processing platform or a cluster framework to work on big data, done by running spark programs, which is run by a help of cluster manager. And with the help of YARN (resource manager) it will start a driver program (master program) creating a DAG which will run operations. Where the YARN runs on JVM (it allows programs to run programs).

So whatever transformation is done it gets saved and the data gets saved in different machines, distributed fashion.

To start session:

1. We need browser
2. IP of load balancer to connect to node of cluster, which will connect to some of worker node

This IP will take you to the kernel.

Type- add credentials

```
jupyter notebook --ip 172.16.0.226  --no browser --port 11552
```

(ip and port is needed, by default port can be common)

Here we are entering one port to get access to the cluster – one way to connect clusters.

3. Winscp – to move files from local to hdfs , vice versa.

# analytics using pyspark

Dataframes : Spark's strongly typed optimized distributed collection of rows.

Note: We create dataframes but spark makes it an RDD backend.

#Follow class 3 and 4 of insofe's spark and sql notebook for further.

BIG DATA

Document by - Shubham kotal