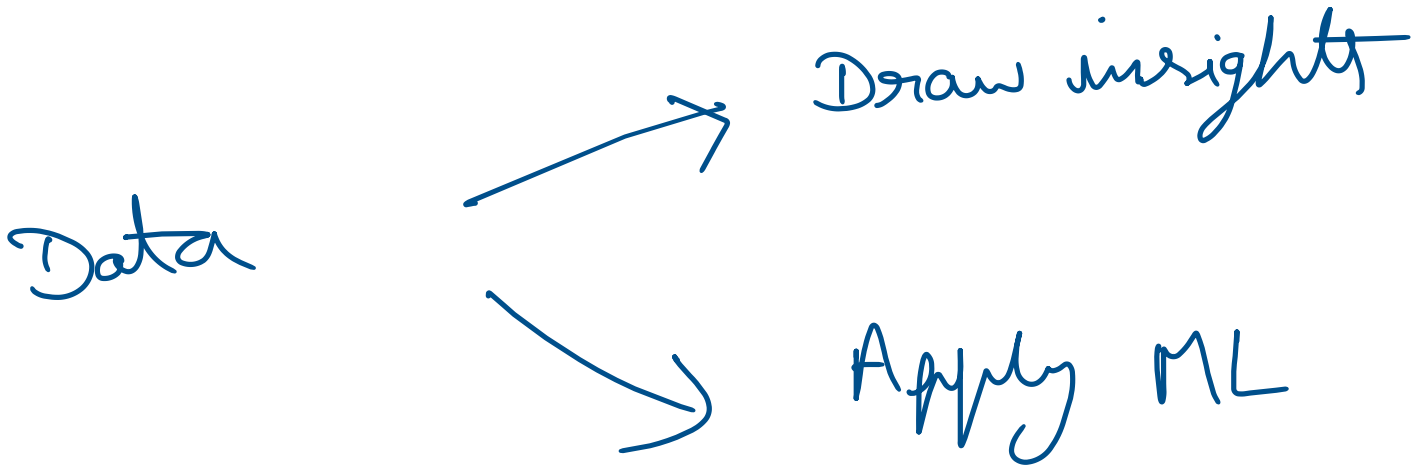
Data Exploration and Analysis with Apache Spark

There is no shortage of data today, Public or Private

Public	Private
Comments	Purchases
Reviews	Page views
Tweets	Clicks
Blogs	Messages
Gov Reports	Surveys
	Questionnaires

untold opportunities.



Doing Something Meaningful with data

- unstructured
- Messy
- Semantically Complex

Data processing

Structured
clean
Easy to consume

Data Processing Task

Parsing fields from text

Accounting for missing values

Identifying and investigating anomalies

Summarizing using tables and charts

Small Data

Somewhat Messy

Spreadsheets

Low data collection frequency

10s – 1000s of rows per day

Sometimes involves manual data collection

Many many files

→ XLSX

Medium Data

High data integrity

Databases

High frequency of collection → SQL

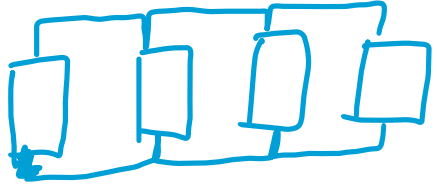
~100K rows per day

programmatically collected

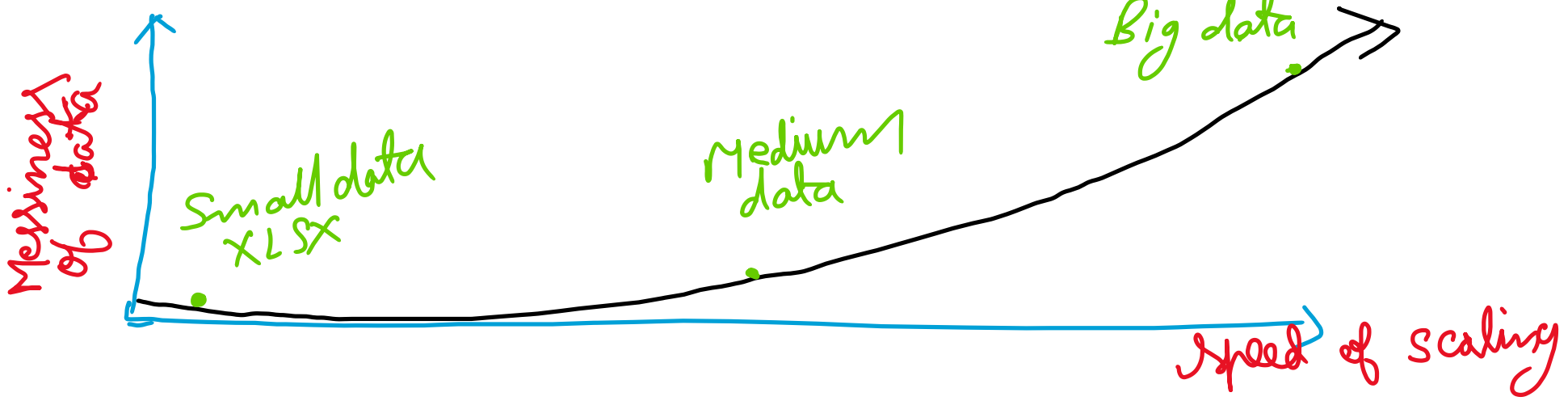
ACID properties

Big Data
very
Merry

Distributed Computing



very high frequency of data collection
millions / Billions of rows per day
files stored across a cluster of machines
many many files (webpages, log files)



Tools for Data Munging

Small data → XLSX

Medium data → SQL, Python, Java, R

Big data → Hadoop Map Reduce, Spark

developed by open source power house Apache

Spark

An engine for data processing and analysis

General Purpose , Interactive , Distributed Computing

General Purpose :-

- Exploring
- Cleaning and preparing
- Apply Machine Learning
- Building Data Applications

Interactive :- Type command and get output

```
len([1,2,5])
>>> 3
```

REPL : Read - Evaluate - Print - Loop

Interactive environment for both Python and Scala

Distributed Computing :- Process data across a cluster of machines

Integrate with Hadoop

To work with Spark, we have Spark API

Spark APIs

Scala

Python

Java

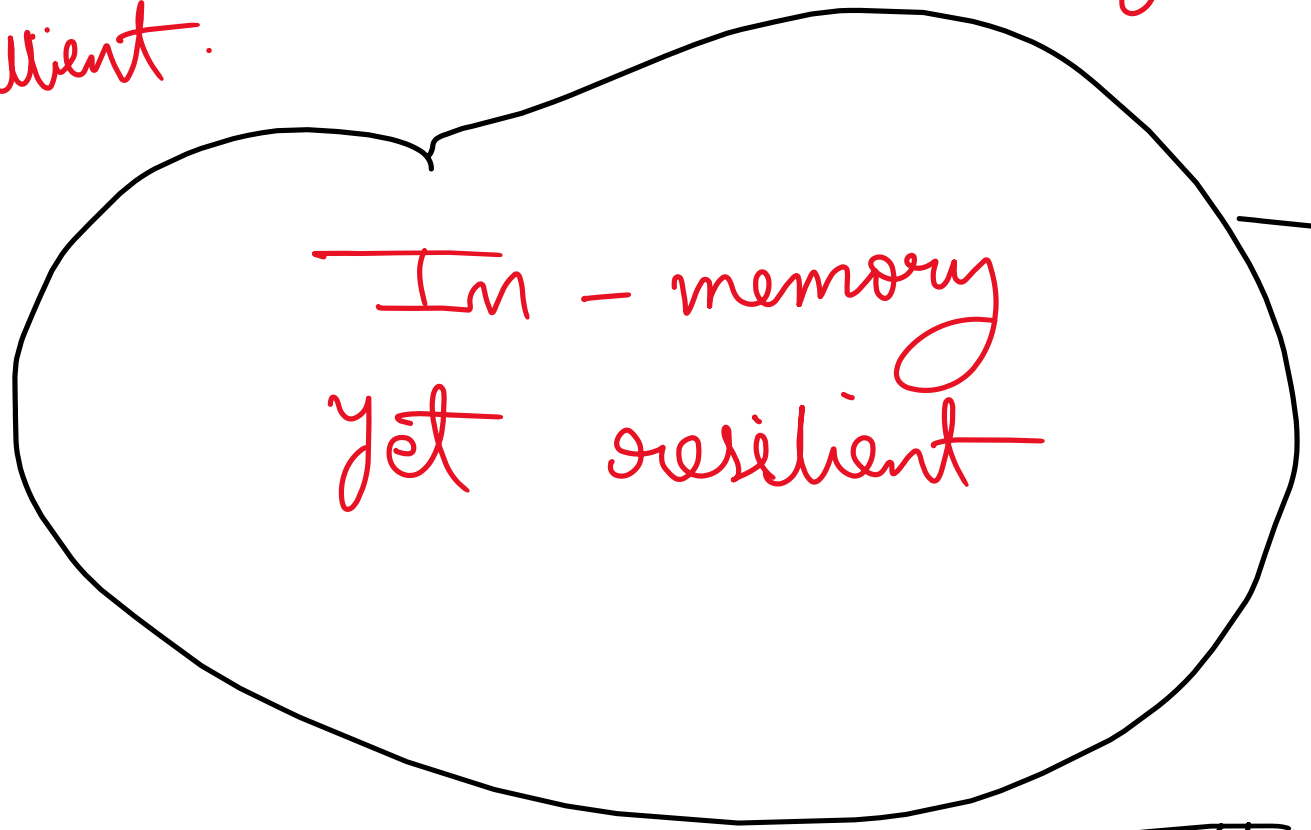
We will be using Spark Python API

Almost all data is processed using

Resilient Distribution Datasets (RDDs)

- RDDs are the main programming abstraction in Spark. They are how spark load data into memory and then processes them.
- RDDs are in-memory collections of objects. Each object in the collection represents one record in your dataset. and all records in the dataset are actually kept in memory across a cluster of machines.

What makes RDDs cool is that they are in-memory, yet they are resilient.



→ resilient
(able to recoil or spring back into shape after bending, stretching or being compressed

They are tolerant to falls or crashes

with RDDs, you can interact and play with billions of rows of data

.... without carrying about any of the complexities involved in resource management or fault tolerance.

Spark is made up of a few different components :—

Spark Core → Handles basic functionality of Spark including creating and processing RDDs.

Spark Core is just a computing engine

Spark needs two additional components because otherwise it is just a computing engine.

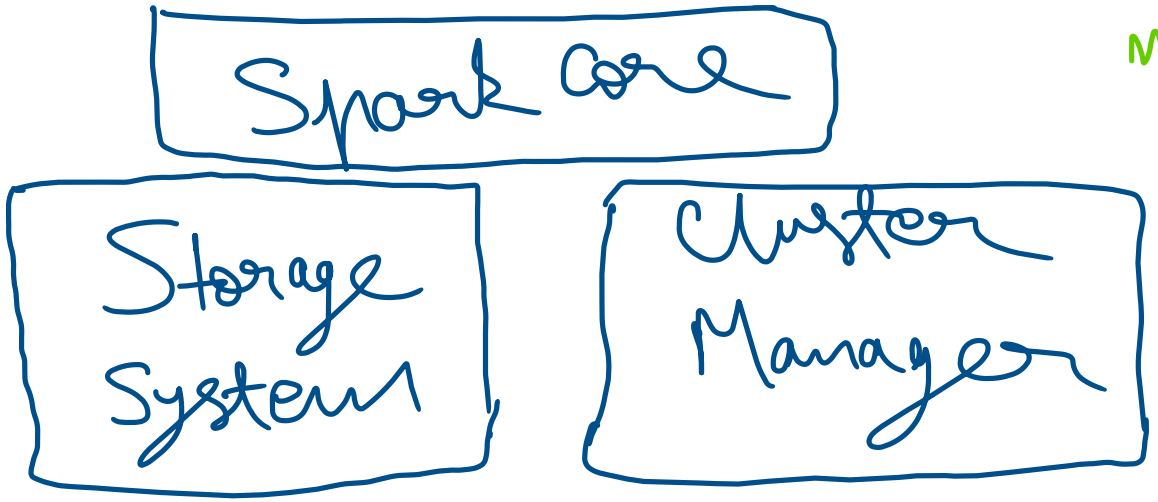
need to Connect with Storage System

Spark Core

a Cluster manager

A Storage System that Stores the data to be processed

A cluster Manager to help Spark run tasks across a cluster of machines



Both of these are Plug and play components

Storage System

: Local file System, used in a standalone mode.

HDFS → which can be plugged in if you are integrating with Hadoop.

Cluster Manager

- Built-in Cluster Manager (comes in the Standalone mode)
- YARN (If you are integrating with Hadoop)

This plug and play capability makes spark really easy to integrate with any existing systems that might be already using Hadoop.

Installing Spark

Prerequisites :- Java 7 or above

Scala

Ipython (Anaconda)

after prerequisities → Go and download spark binaries

update Environment Variables

Spark Environment variables

SPARK_HOME Point to the folder where Spark has been extracted

PATH %SPARK_HOME%/bin

Configure ipython Notebook for Spark

PYSPARK_DRIVER_PYTHON = ipython

PYSPARK_DRIVER_PYTHON_OPTS = "notebook"

Then go to terminal ⟶ pyspark --master local[2]

open Jupyter notebook ↓ check SC Command