Introduction to Big data

Vimala Nunavath

Vimala.Nunavath@uia.no

Outline

- “Hangovers” (from Session 1)
  - the essays
  - the programming projects
- Introduction to big data
- Chapter presentations
  - learning to read and present scholarly work
  - examples of recent research
  - varying difficulty
  - will try to even out

Individual essays

- The essay shall present and discuss selected theory, technology and tools related to big data technologies and EM, backed by scholarly and other references
  - counts 30% of final grade
  - presentations: November 22nd
  - deadline: December 5th 1400
  - send me a brief informal email proposal by next Thursday!
- Encouraged:
  - more than a paper
  - social media contrib's (Wikipedia, Wikidata...)

Some possible Essay Themes:

- Privacy in emergency big data
- Visualization of big data
- Big data ethics.
- Big Data - is it trustworthy?
- Participatory Sensing: A further step. Sensing through Social Media feeds.
- Sentiment Analysis: machine learning approach.
- Redefining communication- the role of social media in disaster management
- Social media and disasters: Current uses, future options, and policy considerations.
- How social media enhances emergency situation awareness?
- Discovering Big data Technologies for EM
- Crisis Analytics: Big Data Driven Crisis Response
- Opportunities and challenges of bid data use in EM?

The student group Programming Project

Group programming project

- The project shall develop an application that can be used for emergency management. Development and run-time platform is free choice, as is programming language. The project should be carried out in groups of three and not more. Working individually or in pairs is not recommended.
- Counts 40% of final grade.
- Final presentation: Friday November 23rd
- Submission deadline: Tuesday November 27th, 1400

Project Topics

1) Integrating different social media applications for providing information awareness either to victims/first responders.
2) Developing an application to detect emerging hot topics over the social media before, during and after emergency management.
3) Social media data analytics for disaster management.
4) Deep Learning for emergency management Using Social Media Information.
5) Analysis of Post-disaster Twitter Communication: A case study of….
6) Develop an information visualization tool to identify the disaster risk.

Integrating different social media data sources for Information awareness.

- Developing an application for integrating different social media data sources for information awareness.

Developing an application to detect emerging hot topics over the social media before, during and after emergency management.

- Developing an application for detecting the disaster trends in various regions.

Social media data analytics for emergency management

- Develop a system capable of processing, analyzing, and extracting useful information from Twitter during an emergency to understand the public sentiment.

  e.g. Usecase: Kerala floods 2018.

Deep Learning for emergency management Using Social Media Information.

- Classification of social media information by using different machine learning algorithms.

Analysis of Post-disaster Twitter Communication: A case study

- Develop an application to understand/classify the communication tasks that are performed by using Twitter during post–disaster phase.

Information visualization tool to identify the disaster risk.

- Develop an information visualization tool to identify the disaster risk.

Should we find a coordinated project task?

Big Data:

- Popular since late 2000's
  - buzzword, over-hyped, maybe already waning
  - but there is a (disruptive) reality behind it:
    - ever increasing amounts of available data
    - go beyond capabilities/capacities of established computing techniques and tools
    - calls for new understandings, techniques and tools
- Our working definition for now:

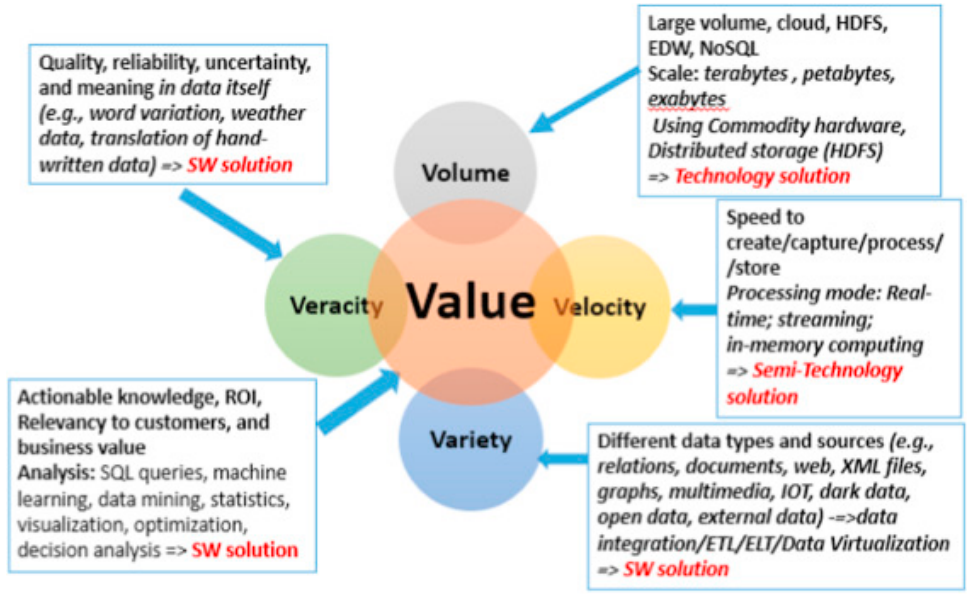
  "the ever increasing amount of available data today that go beyond the capabilities/capacities of existing solutions and thus calls for new understandings, techniques and tools".

Based on chapters 4 and 5 in Kitchin:

- Chapter 1 is about data
- Chapter 2 is about small data, infrastructures and brokers
- Chapter 3 is about open and linked data
- And we may get back to (some of) it
  - but we prefer to jump right into the most central themes from the start
  - chapters 4 and 5 are quite possible to read without the ones before

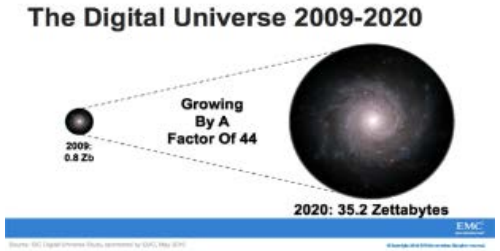
Characteristics of Big data:

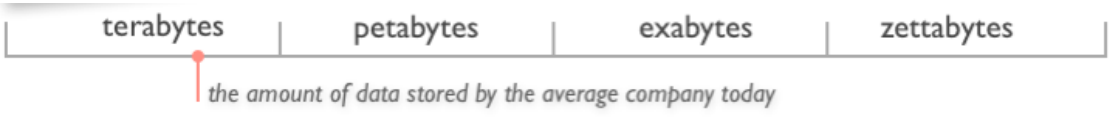
- The “three V's” (3V):
  - volume, velocity, variety – at once
    - old days: you could only have two of the three
  - also two more: veracity, value
- Other characteristics:
  - exhaustive in scope: “n = all”
  - fine-grained in resolution
  - indexical
  - relational in nature
  - flexible: extensional
  - flexible: scalable

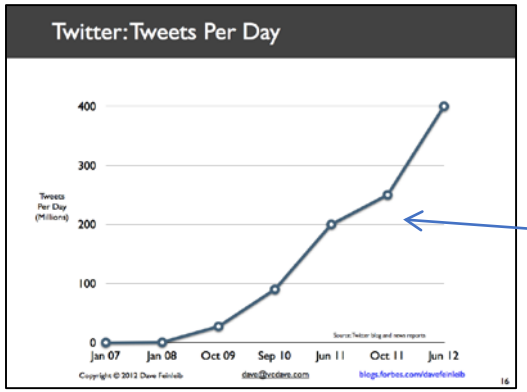


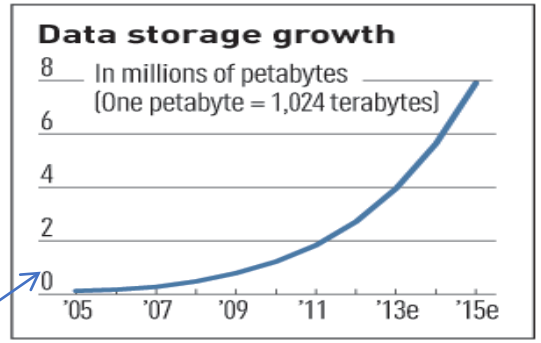
Volume:

- **Data Volume**
  - 44x increase from 2009 2020
  - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially









Exponential increase in collected/generated data

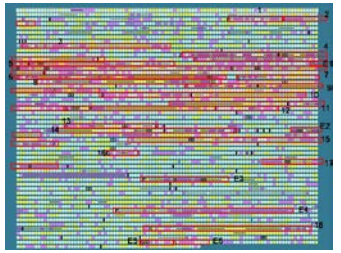
Velocity:

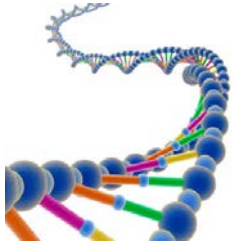
- Velocity:
  - created rapidly, in or near real time
  - analysis on the fly, not always storing it all

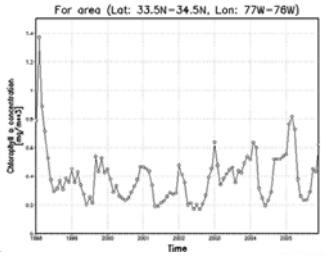
- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities
- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction

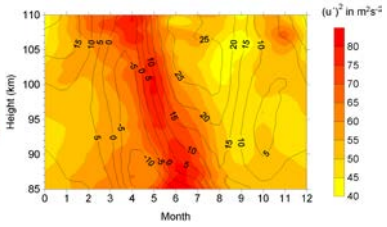
Variety:

- Relational Data (Tables/Transaction/Legacy Data)
- Text Data (Web)
- Semi-structured Data (XML)
- Graph Data
  - Social Network, Semantic Web (RDF), …
- Streaming Data
  - You can only scan the data once
- A single application can be generating/collecting many types of data
- Big Public Data (online, weather, finance, etc)
- Some temporal, some spatial, some both, some neither – some socially networked, some thematically grouped







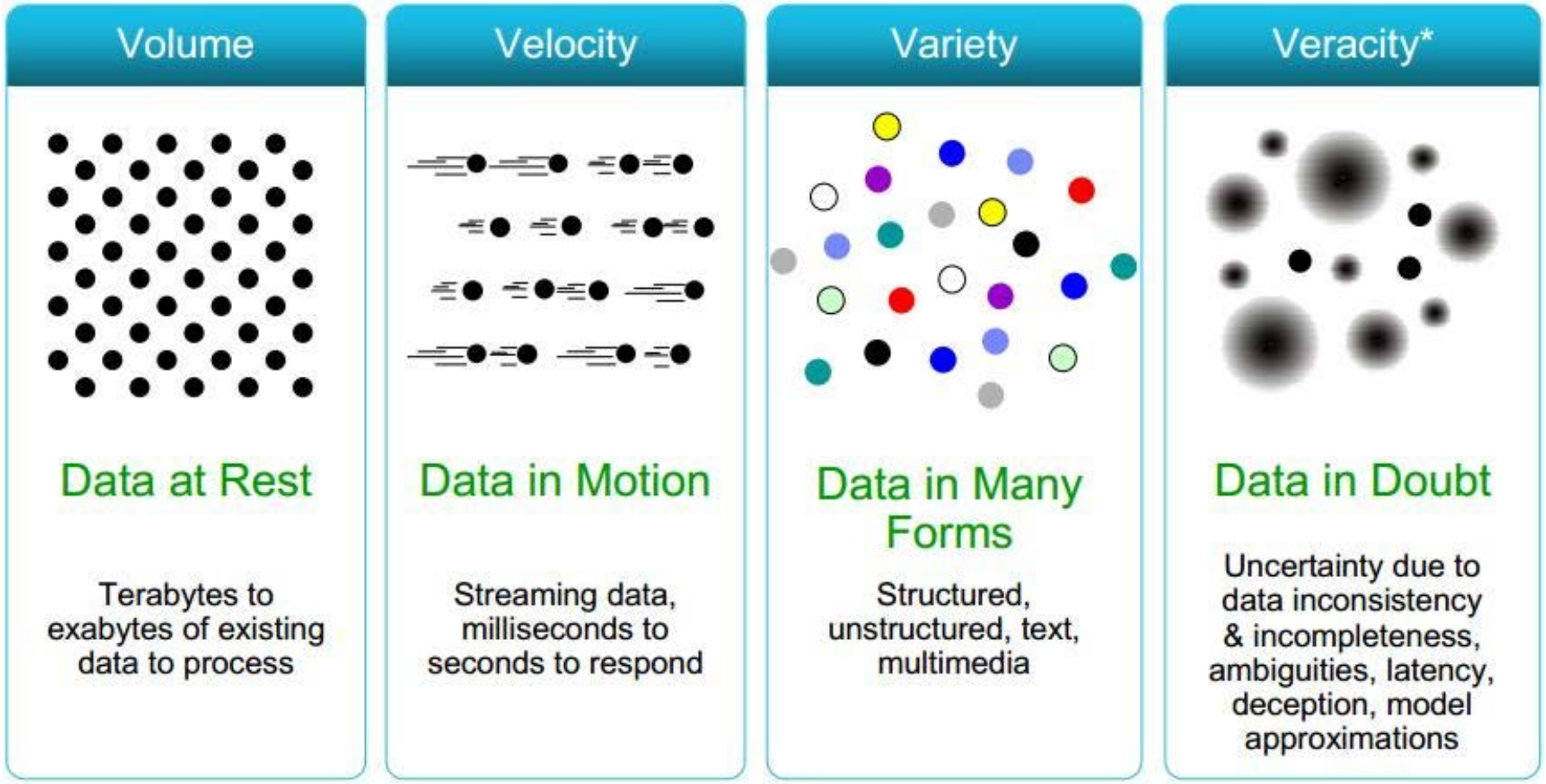


To extract knowledge➔ all these types of data need to linked together

Veracity and Value:

- Veracity:
  - the trustworthiness of data: quality
    - accuracy, correctness, provenance
  - big data quality is uneven and can be low
    - e.g., microblog streams
  - how and when can volume make up for quality?
- Value
  - how to make value out of the data?
    - both commercial and societal
  - e.g.: understand/serve customers/citizens; optimize business processes; “nowcasting”; assess teaching effectiveness; societal safety; detect cyber crime...

Some make it 4V's



Exhaustiveness, resolution, indexicality:

- Exhaustiveness
  - capturing and analyzing data about everyone/-thing
    - instead of sampling
- Fine-grainedness in resolution
  - aiming to be as fine-grained as possible
  - collecting, storing and analyzing smallest data points
    - instead of storing aggregate values
- Indexicality
  - unique identifiers for everyone and everything
    - trying to match different identifiers for the same person or thing (e.g., user names/handles)
  - using IRIs to identify resources on the Web of Data

Relationality:

- People and things are described in ways that make them combinable with
  - other related persons and things
  - other descriptions of the same persons and things

Flexibility:

- Extensionality
  - easy to add new data to the data set
- Scalability:
  - big datasets should be able to scale rapidly
  - use of grid computing, cloud servers, NOSQL databases (Not-Only SQL)

Enablers of big data:

- Computation
  - "Moore's law" of transistor numbers (1965 –)
- Networking
  - "Gilder's law" of network bandwidth (2000 –): global bandwith doubles every 6 months
- Storage (cloud, *aaS, NOSQL)
- Pervasive and ubiquitous computing
  - sensors and actuators
  - from dumb to smart things (cars)
  - exhaustive data collection
  - "ambient computing", "the age of everyware"

Pervasive versus ubiquitous computing:

- Pervasive computing:
  - computing “in everything”
  - make them interactive and smart
  - divergent: more and more things become smart
  - needs situational awareness
- Ubiquitous computing:
  - computing “in every place”
  - moves with the person
  - convergent: smart things we carry do more and more
  - needs context and location awareness

Enablers of big data:

- Indexicality
  - growth of unique identifiers
  - people: user names/handles, personal numbers / SSNs, passports, driver's licenses, health cards, biometry, IMSIs
  - things (and information): product type codes, RFID for individual products, auto passes, MAC addresses, IMEIs, IRIs, including ISBNs, ISSNs, DOIs, etc.
  - places: post codes, addresses, geo coordinates
- Machine-readable identification
  - more and more are becoming digital
- ...and remote readable

Sources of big data:

- Three types:
  - directed
  - automated
  - volunteered
- Directed data collection
  - organized and structured surveillance
  - personal or through technological lens
  - census, government forms, inspections, CCTV cams
  - surveillance technology is becoming digital, smarter, directable, internetworked...

Automated data collection:

- Automated surveillance
  - e.g., smart electricity meters, electronic transportation tickets, passenger counting systems, car tolls, radar/LiDAR speed guns, ANPR
- Digital devices
  - smart phones/tablets + lots of others
  - actively produce data
  - primary: cameras, videos, GPS units, medical devices
  - exhaust: mobile phones (also primary), cable boxes
  - logjects = objects that log their (+ their users') history
  - objects can also be logged by others • e.g., mobile-device triangulation

Automated data collection:

- Interaction data
  - all ICT-based transactions leave traces
  - using a web shop, net bank, ATM
  - sending an email
  - accessing the internet from home or a mobile device
- Scan data
  - machine-readable identification codes
  - barcodes, QR (“Quick Response”) codes
  - magnetic cards, chip card/smart card/ICC
- Sensor (sensed) data
  - inexpensive sensor generate continuous data streams
  - smart cities gauging noise, temperature, light, CO2 ...

Volunteered data collection:

- Social media, collective projects (online)
  - production + consumption = prosumption
- Transactions
  - voluntary registration, clickstreams, review data
- Some of the automated collection was volunteered:
  - actively produced data
  - primary: cameras, videos, GPS units, medical devices
  - some logjects (objects that log history)
- Sousveillance
  - (fr.) sur-: above, sous-: below
  - self-monitoring, e.g., wearable fitness equipment, dieting apps

Volunteered data collection:

- Social media
- Crowdsourcing
  - to create one new product
  - to create many new products/concepts/ideas
  - to assess many existing products/concepts/ideas
- Citizen science

Big Data as a Disruption:

- Disruptive technology:
  - a technology that displaces established ones, and shakes up existing or creates new industries
  - e.g., PCs, the internet, digital media, social media
- Big data is disruptive
  - it creates new data-driven organization forms
  - new ways of doing research and science
  - new ways of creating and maintaining products and services
  - new threats to privacy and social order
- ...too easy to shrug off (just) as a hype/buzzword

Data-driven organizations:

- "The next phase of the knowledge economy, reshaping the mode of production" (RK, p. 16)
  - inward: monitor, evaluate performance in real time; reduce waste and fraud; improve strategy, planning and decision making
  - outward: design new commodities, identify and target new markets, implement dynamic pricing, realise untapped potential, gain competitive advantage
- Goals: run more intelligently; flexibility and innovation; reduced risk, cost, losses; improved customer exper., return on investment, profit

New ways of doing business:

- Marts (Walmart, Kohl's): analyze sales, pricing, economic, demographic and weather data to tailor local product selection and price markdowns
- Online dating: sift through personal characteristics, reactions and communications to improve matches
- NY Police: analyze data on past arrests, paydays, sporting events, weather and holidays to deploy officers optimally
- Professional sports: massaging sports statistics to spot undervalued players
- Education: analyze data from learning management systems to improve teaching / studying

Steve Lohr (2012): The Age of Big Data, NYTimes.com

What is Spark?

Fast and **Expressive** Cluster Computing Engine Compatible with Apache Hadoop

Up to **10×** faster on disk, **100×** in memory

2-5× less code

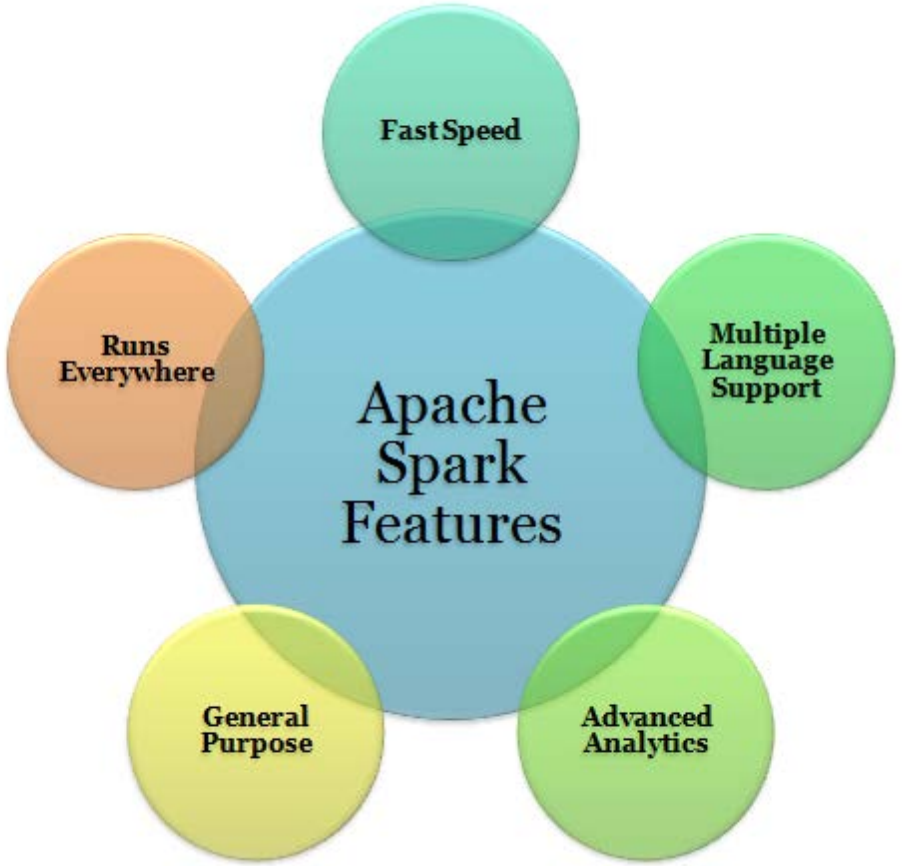
Efficient

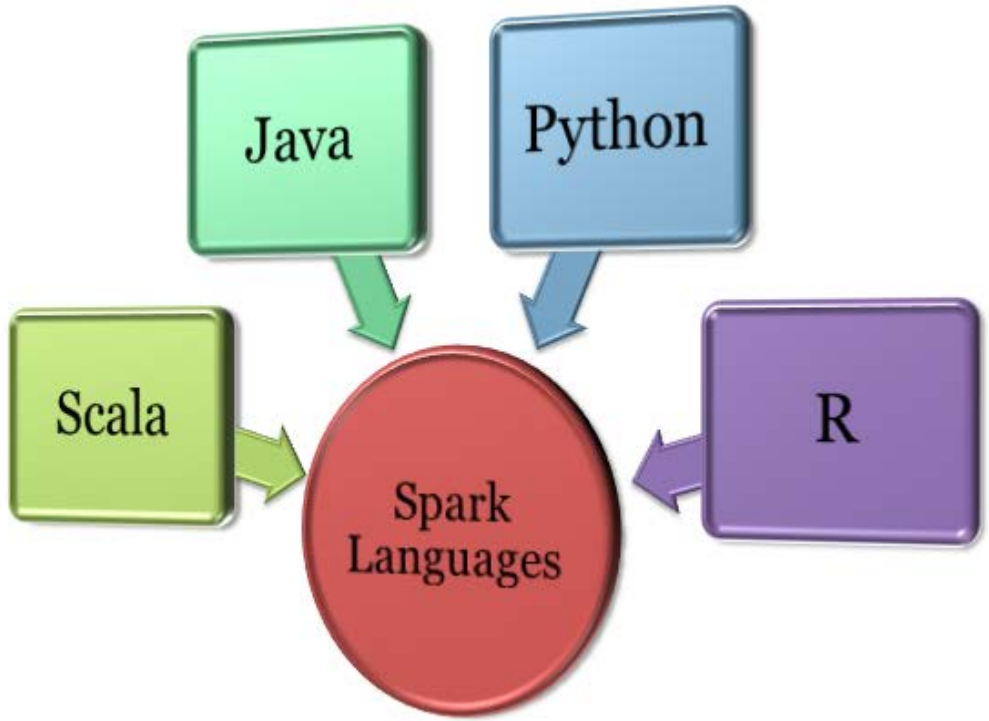
- General execution graphs
- In-memory storage

Usable

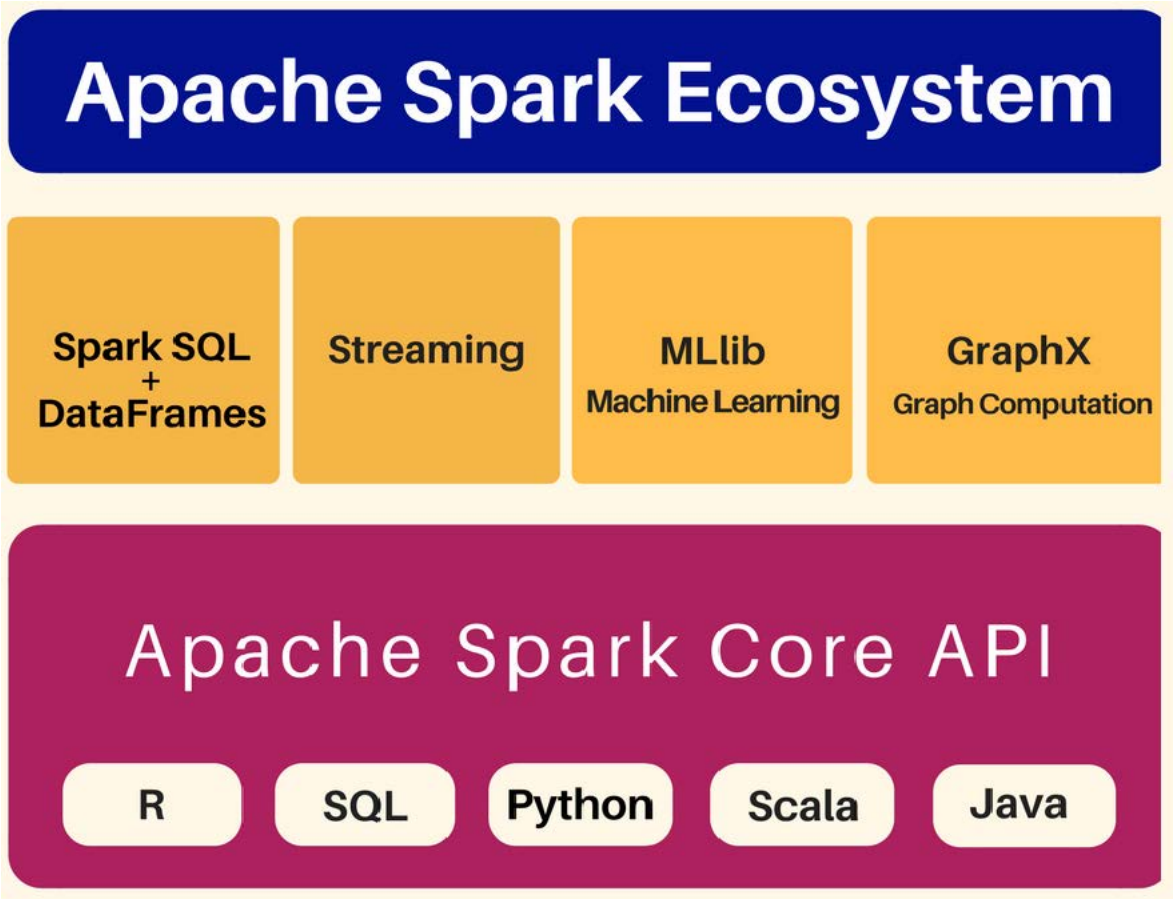
- Rich APIs in Java, Scala, Python
- Interactive shell

Apache Spark Features





Components of Apache Spark Ecosystem



Spark Core

- The main execution engine of the Spark platform is known as Spark Core.
- All the working and functionality of Apache Spark depends on the Spark Core including memory management, task scheduling, fault recovery, and others.
- It enables in-memory processing and referencing of big data in the external storage systems.
- It is responsible to define RDD (Resilient Distributed Dataset) by an API that is the programming abstraction of Spark.

Spark SQL and DataFrames

- the main component of Spark that works with the structured data and supports structured data processing.
- comes with a programming abstraction known as DataFrames.
- performs the query on data through SQL and HQL (Hive Query Language, Apache Hive version of SQL).
- enables developers to combine SQL queries with manipulated programmatic data that are supported by RDDs in different languages.
- This integration of SQL with advanced computing medium combines SQL with the complex analytics.

Spark Streaming

- It is responsible for the live stream data processing such as log files created by production web servers.
- It provides API for the manipulation of data streams, thus makes it easy to learn Apache Spark project.
- It also helps to switch from one application to another that performs manipulation of real time as well as stored data.
- This component is also responsible for throughput, scalability, and fault tolerance as that of the Spark Core.

MLlib

- It is the in-built library of Spark that contains the functionality of Machine Learning, known as MLlib.
- It provides various ML algorithms such as clustering, classification, regression, collaborative filtering and supporting functionality.
- MLlib also contains many low-level machine learning primitives.
- Spark MLlib is 9 times faster than the Hadoop disk-based version of Apache Mahout.

GraphX

- the library that enables graph computations.
- also provides an API to perform graph computation by allowing users generate directed graph using arbitrary properties of the edge and vertex.
- Along with the library for manipulating graphs, it provides many operators for the graph computation.

Why Apache Spark?

- Ease of use
- High-performance gains
- Advanced analytics
- Real-time data streaming
- Ease of deployment

Resilient distributed dataset (RDD):

- which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel.
- either transform data or take actions on that data.

Key Concept: RDD's

Write programs in terms of **operations** on distributed datasets

Resilient Distributed Datasets

- Collections of objects spread across a cluster, stored in RAM or on Disk
- Built through parallel transformations
- Automatically rebuilt on failure

Operations

- Transformations (e.g. map, filter, groupBy)
- Actions (e.g. count, collect, save)

Transformations:

Transformations

The following table lists some of the common transformations supported by Spark. Refer to the RDD API doc (Scala, Java, Python, R) and pair RDD functions doc (Scala, Java) for details.

Transformation	Meaning
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.
flatMap(*func*)	Similar to map, but each input item can be mapped to 0 or more output items (so *func* should return a Seq rather than a single item).
mapPartitions(*func*)	Similar to map, but runs separately on each partition (block) of the RDD, so *func* must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.
mapPartitionsWithIndex(*func*)	Similar to mapPartitions, but also provides *func* with an integer value representing the index of the partition, so *func* must be of type (Int, Iterator<T>) => Iterator<U> when running on an RDD of type T.
sample(*withReplacement*, *fraction*, *seed*)	Sample a fraction *fraction* of the data, with or without replacement, using a given random number generator seed.
union(*otherDataset*)	Return a new dataset that contains the union of the elements in the source dataset and the argument.
intersection(*otherDataset*)	Return a new RDD that contains the intersection of elements in the source dataset and the argument.
distinct([*numPartitions*]))	Return a new dataset that contains the distinct elements of the source dataset.

Actions:

Action	Meaning
reduce(*func*)	Aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	Return the number of elements in the dataset.
first()	Return the first element of the dataset (similar to take(1)).
take(*n*)	Return an array with the first *n* elements of the dataset.
takeSample(*withReplacement*, *num*, [*seed*])	Return an array with a random sample of *num* elements of the dataset, with or without replacement, optionally pre-specifying a random number generator seed.
takeOrdered(*n*, *[ordering]*)	Return the first *n* elements of the RDD using either their natural order or a custom comparator.

- *What to do in Two Weeks?*

...and in the meantime :-)