ST446 Distributed Computing for Big Data

Lecture 1

Course Overview

Milan Vojnovic

https://github.com/lse-st446/lectures2021

Topics of this lecture

- Course organization and resources
- Overview of lecture/seminar topics
- Information for Seminar class 1

Schedule of lectures and seminars

- Lectures:
  - Video recordings available on Moodle
  - Office hours: Wednesday 16:00-17:00 (via Zoom)

- Seminars:
  - Group 1: Thursday 09:00-10:30
  - Group 2: Friday 12:30-14:00

  - Office hours (week 2 onwards):
    - Tuesday 14:00-15:00
    - TBD

Your team

- Lectures:

  Milan Vojnovic, Department of Statistics, m.vojnovic@lse.ac.uk

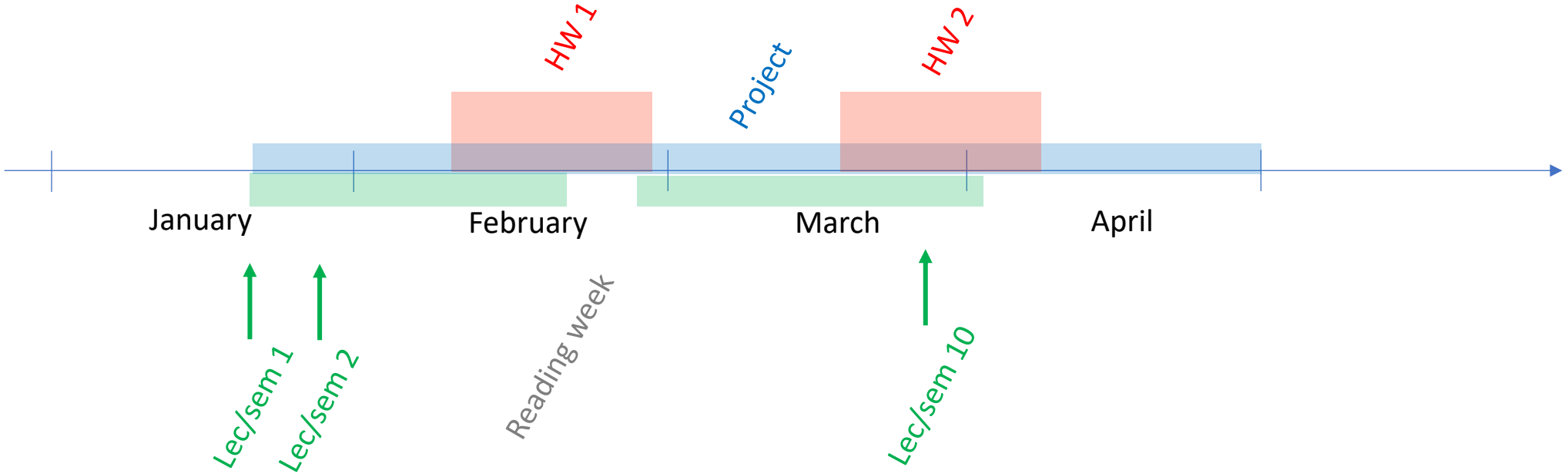
- Seminars:

  Marcos Barreto, Department of Statistics, m.e.barreto@lse.ac.uk

  Jenjira Jaimunk, Department of Statistics, jenjira.jaimunk@kcl.ac.uk

  Kaifang Zhou, Department of Statistics, k.zhou3@lse.ac.uk

Course workflow

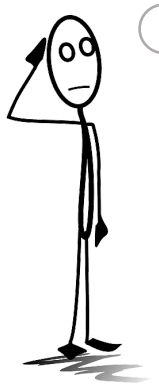


What is this course about?

- Introduction to distributed systems for data processing
- Introduction to data models, programming paradigms, and system architecture
- Hands on approach: concepts taught in lectures implemented in seminar sessions

What is this course not about?

- A course focused solely on API (Application Programming Interface)
  - We will cover some principles of computer system architecture
- A course focused on principles of distributed computing
  - Though we will cover some principles
- A course on theory of computation
  - This is not a course focused solely on algorithms for parallel computation

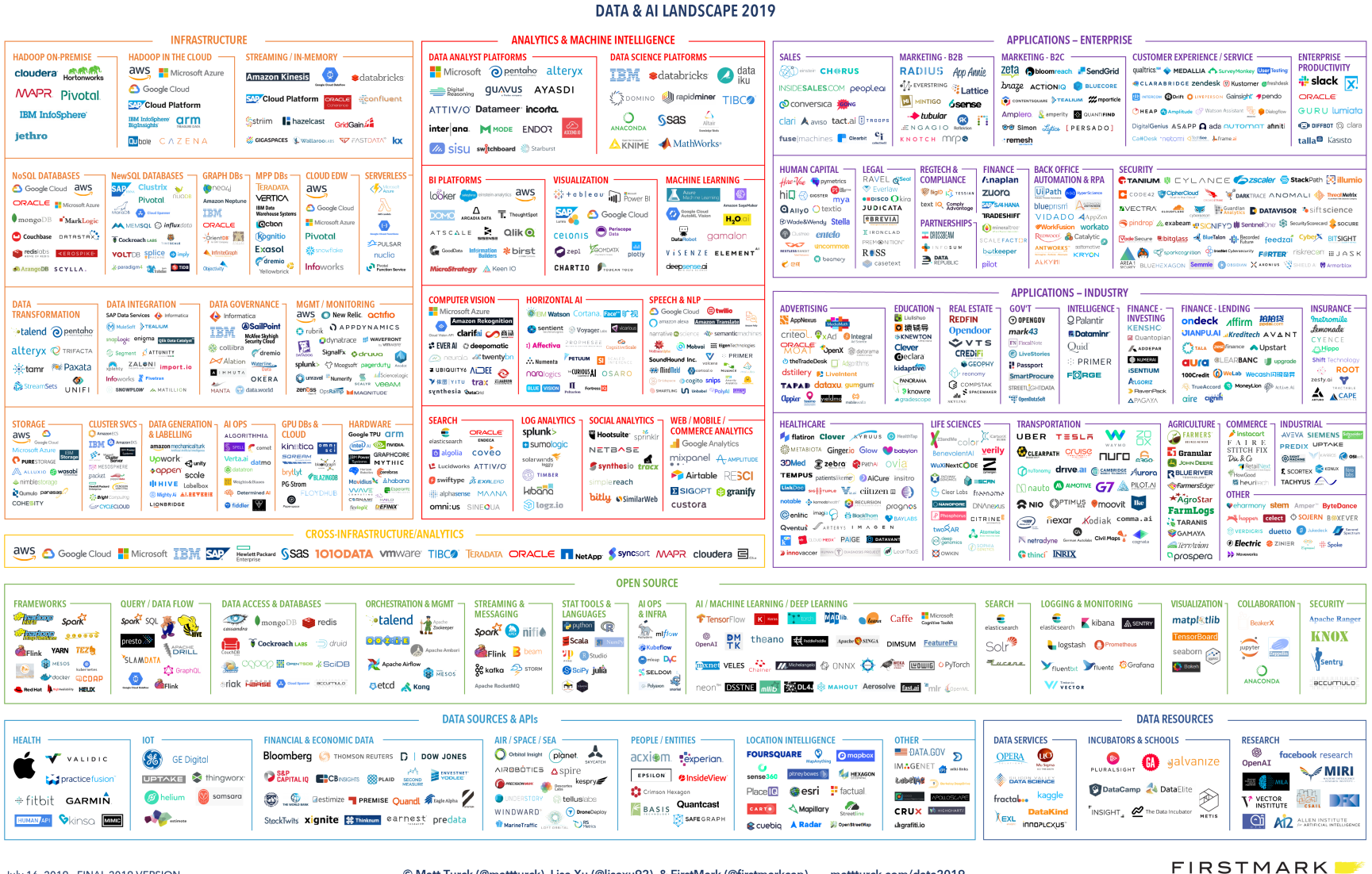


Hmm … this seems to be a computer science type of a course … such courses should not be taught at LSE

… this course is probably not a good choice for you

Course outline

1. Introduction
2. Distributed file systems and key-stores
3. Distributed computation models
4. Structured queries over large datasets
5. Graph data processing
6. Stream data processing
7. Distributed optimization methods for machine learning
8. Machine learning optimization for matrix factorization and topic modelling
9. Scaling up distributed machine learning
10. Distributed dataflow graph computations



- There are many existing data processing systems
- The course should help you to learn basic principles and about some main systems

Software used in this course

- Python
- Jupyter notebooks
- Hadoop
- Spark https://spark.apache.org/
  - A unified analytics engine for large-scale data processing
- Google Cloud Platform https://console.cloud.google.com/
  - Bigtable, BigQuery, Dataproc, TensorFlow, …
  - Sponsored credits: USD 50.00 per student

Resources

- Course handout: https://lse-st446.github.io
  - Short summary of topics covered in each week of the course

- GitHub lse-st446 organization: https://github.com/lse-st446
  - Contains lecture and seminar materials
  - Homework assignments (GitHub classroom)
  - Project repositories

- Some useful links:
  - Lectures: https://github.com/lse-st446/lectures2020
  - Resources: https://github.com/lse-st446/lectures2020/blob/master/Resources.md
  - Projects: https://github.com/lse-st446/lectures2020/blob/master/Projects.md

GitHub: you need to join ST446 org

- Send us your GitHub account name by filling this Google form:

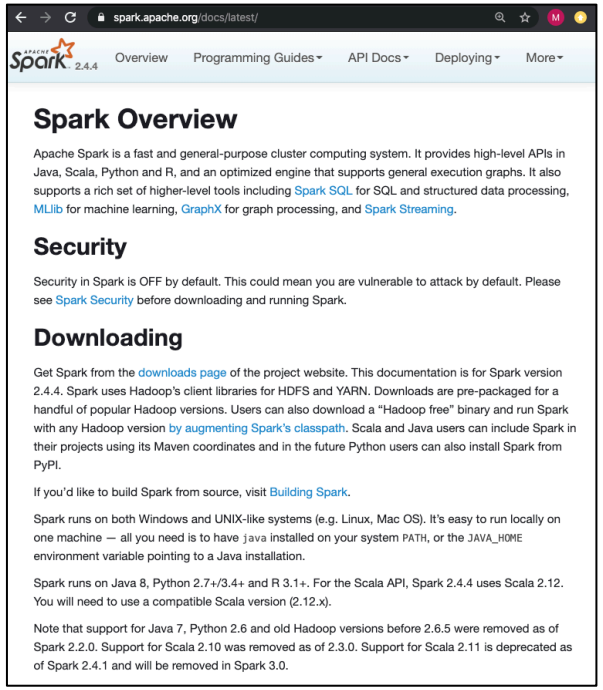
  https://forms.gle/DSnvVfQPqBiyXHZe9 [update link]

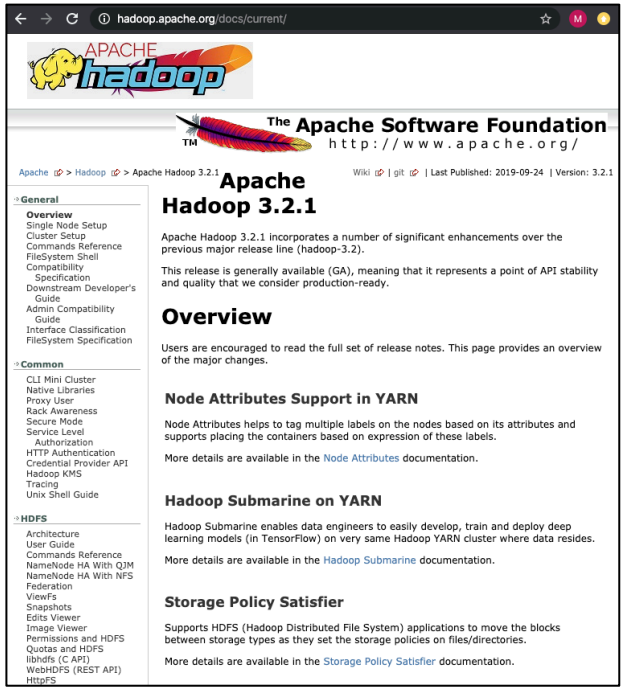
- Do this before the first seminar session !

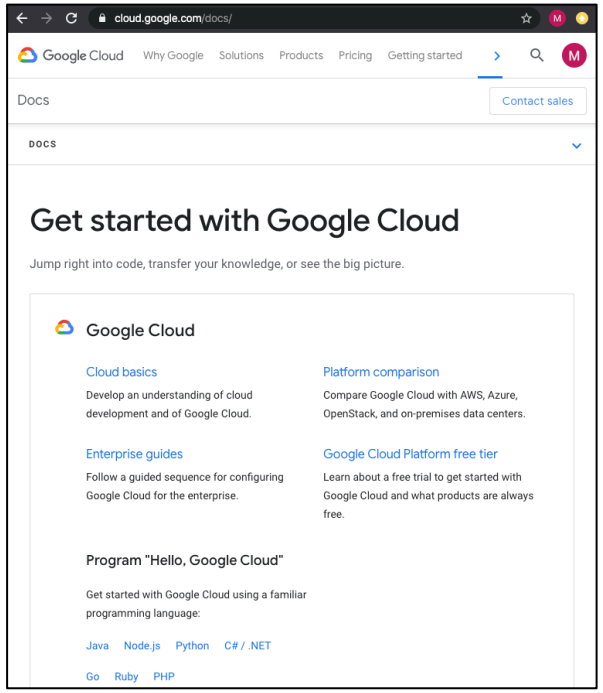
- If you don't have a GitHub account, it is easy to create one: http://github.com

- You **must** use a GitHub account that **does not reveal your identity** (if enrolled) **!!!**

Resources: online documentation



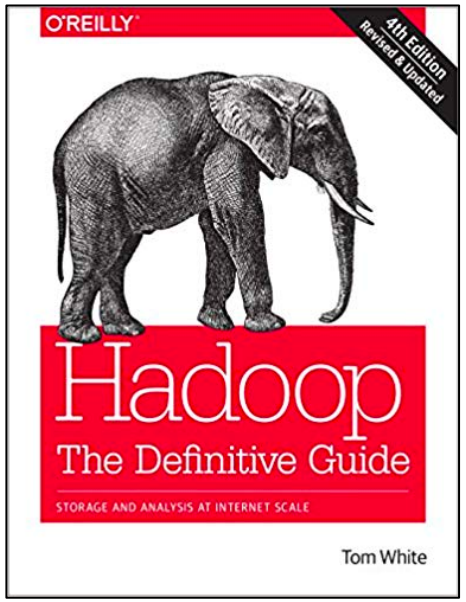


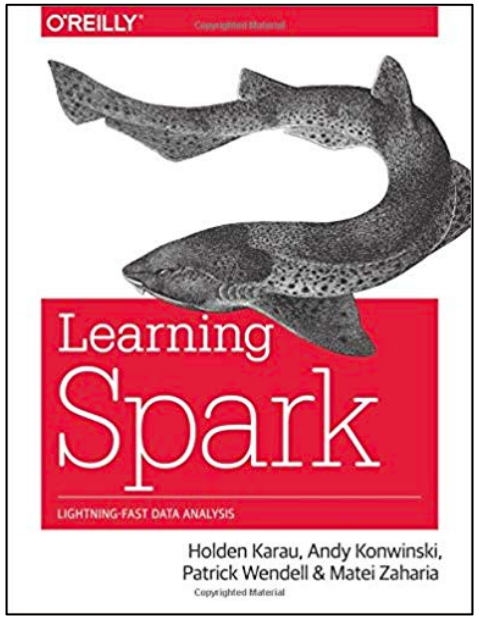


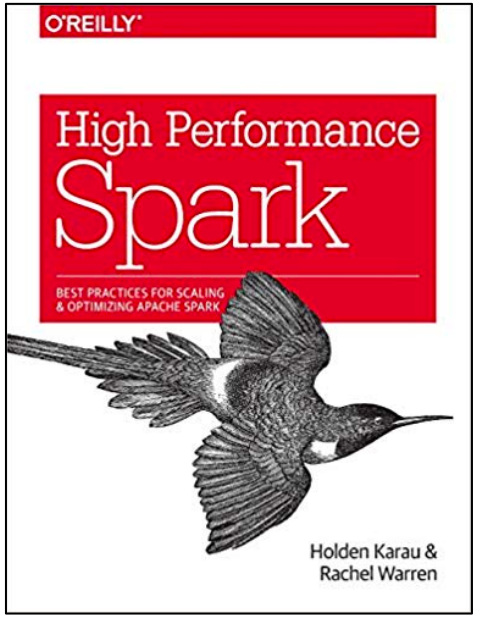
https://spark.apache.org/docs/latest/ https://hadoop.apache.org/docs/current/ https://cloud.google.com/docs/

Additional resources

- No book covers all topics of this course
- There are many books on different computing systems







...

Additional resources

- Research papers
  - Papers published in computer systems conferences (SOSP, NSDI, SIGMOD, …)
  - Many existing systems were first developed for production systems of large companies such as Amazon, Google, and Microsoft
    - Their architecture is commonly described in research papers
    - Typically followed by open-source implementations
  - Active area: new systems are continuously being developed
- DataCamp (link on GitHub – see Resources.md)

Computer systems conferences

- ACM Symposium on Operating Systems Principles
  - SOSP
- USENIX Symposium on Networked Systems Design and Implementation
  - NSDI 2020
- IEEE International Conference on Data Engineering
  - ICDE 2019
- USENIX Annual Technical Conference
  - USENIX ATC 2020
- ACM Special Interest group on Management of Data
  - SIGMOD 2019
- International Conference on Very Large Databases
  - VLDB 2019

Evaluation and marking

- Continual assessment
  - Two assignments each 10% of the mark
  - First assignment in lecture #4
  - Second assignment in lecture #9

- Course project
  - 80% of the mark
  - Project topic discussions with the lecturer

- Important dates:
  - **Feb 27** first assignment solution
  - **Apr 10** second assignment solution
  - **Apr 30** project report

Course projects

- Individual project on a topic within the scope of the course
  - Focus on methodological aspects, principles, and implementation
- Allowing for
  - Accounting for individual preferences
  - Exploring a given topic in depth
  - Gaining a hands-on experience in coding a solution
- Some hints for project topics:
  - Analyze a dataset using a structured query language
  - Analyze a large dataset using Spark's RDD or Dataframe API
  - Train a classifier using a large training dataset

  - More hints in the GitHub lse-st446 file Projects.md

Introduction to big data and distributed computing systems

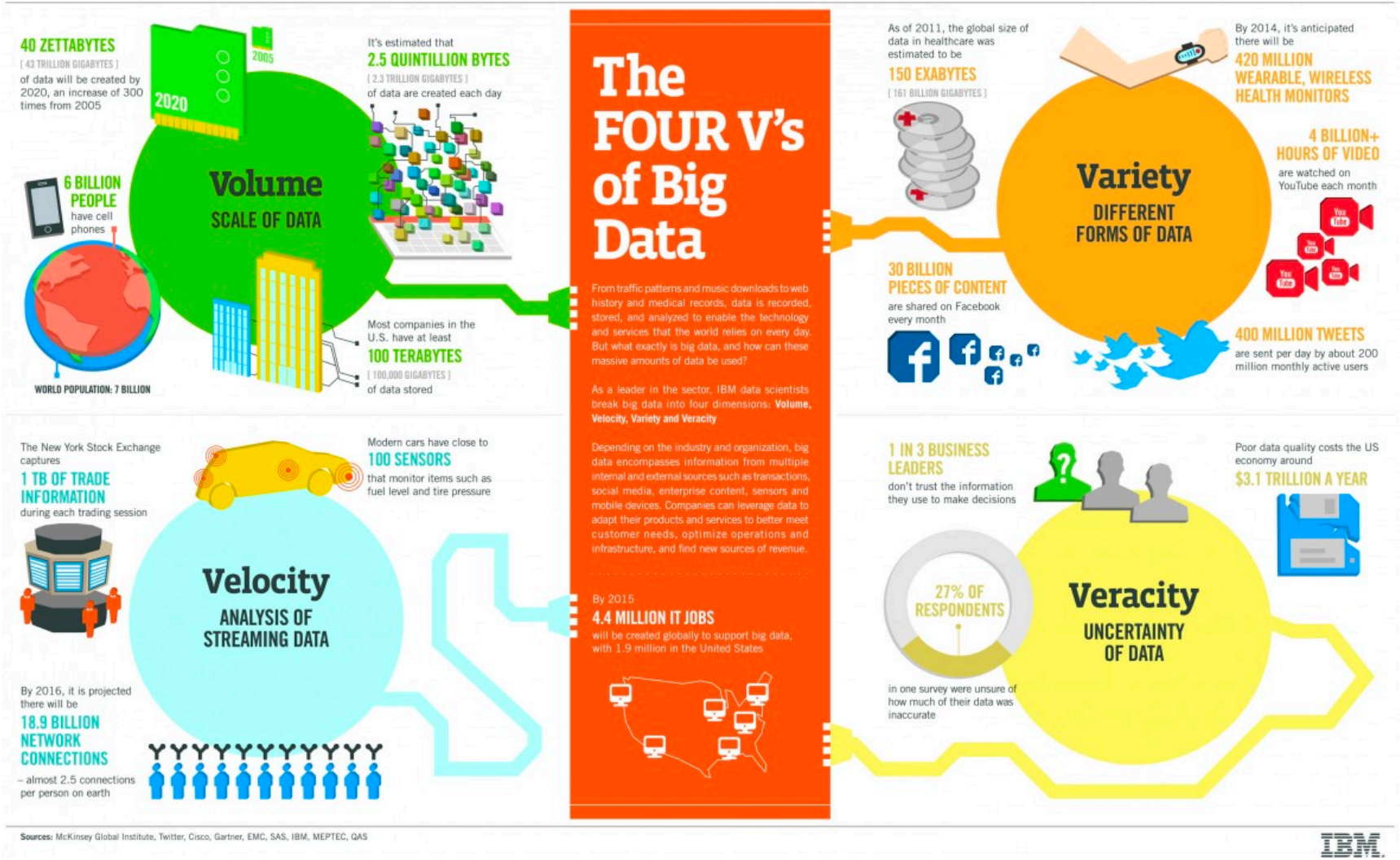
Big data

- The term “big data” in use since 1990s
  - No single universally accepted definition

- Our working definition: a dataset whose volume or complexity make traditional data processing application software inadequate

- “Big data” has at least one of the following properties:
  - Volume: large quantity of generated and stored data
  - Variety: data may be of different type
  - Velocity: data may be generated at different speeds
  - Veracity: the quality of captured data can vary greatly

The four V's of big data



Data volume sizes

Unit	Abbreviation	Total bytes	Nearest decimal
kilobyte	KB	1024	10^3
megabyte	MB	1024^2	10^6
gigabyte	GB	1024^3	10^7
terabyte	TB	1024^4	10^8
petabyte	PB	1024^5	10^9
exabyte	EB	1024^6	10^{10}
zettabyte	ZB	1024^7	10^{11}
yottabyte	YB	1024^8	10^{12}

1 B = 8 bits
1 KB = 2^{10} B
kB vs KB see here https://en.wikipedia.org/wiki/Kilobyte

Examples of data volumes

Human population
7.6B people

Mobile phones
4.77B people

Web
4.56B pages

Facebook
2.07B monthly active users
1.37B daily active users
300M photo uploads per day
338 friends per user

Wikipedia
44M total wiki pages
6M English articles
10TB uncompressed
100GB compressed
(7-zip, with complete edit History in XML format, 2015)

Google knowledge graph
70M facts

Yago knowledge base
10M+ entities
120M facts

Linkedin
500M users

Twitter
328M monthly active users

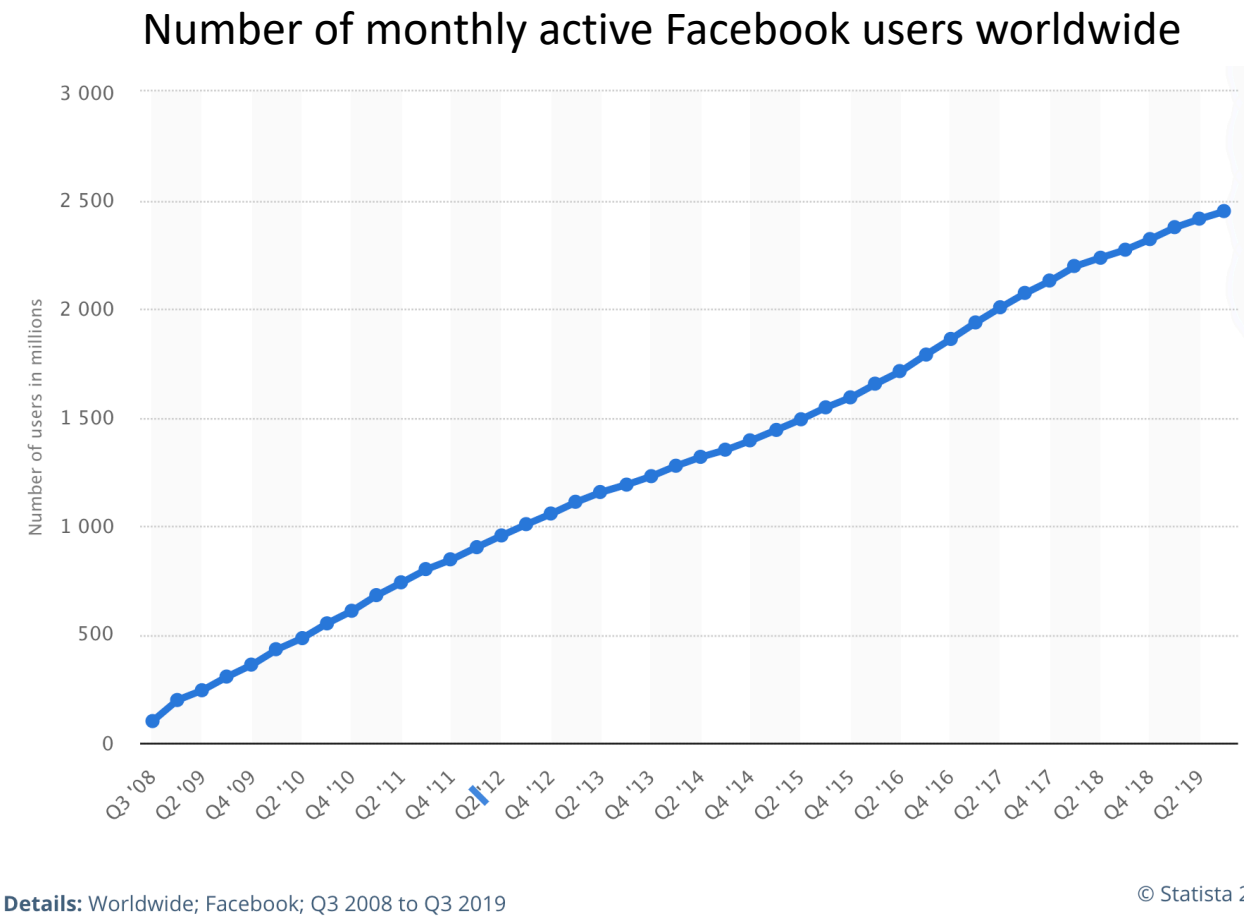
Criteo terabyte click logs
50GB raw data per day
195M ad impressions per day
26 features

Imagenet
14M+ photos
1.2TB compressed size

Sources

- Human population: http://www.worldometers.info/world-population/
- Mobile phones: https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/
- Wikipedia: https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia
- Web: http://www.worldwidewebsize.com/
- Facebook:
  - https://zephoria.com/top-15-valuable-facebook-statistics/
  - https://www.brandwatch.com/blog/47-facebook-statistics-2016/
  - https://techcrunch.com/2012/08/22/how-big-is-facebooks-data-2-5-billion-pieces-of-content-and-500-terabytes-ingested-every-day/
- Linkedin: http://fortune.com/2017/04/24/linkedin-users/
- Google knowledge graph: https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html
- Yago: https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/#c10444
- Twitter: statista https://www.statista.com/topics/737/twitter/
- Criteo: http://labs.criteo.com/2013/12/download-terabyte-click-logs/
- ImageNet: http://image-net.org/

Data volume is ever increasing



- https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/
- Growing data: need data processing systems that can scale !

Data source types

- Structured: any data that has a schema, i.e. a known set of fields for each record
  - Relational Database Management Systems (RDBMS)
  - Ex. Oracle database, IBM DB2, Microsoft SQL Server

- Semi-structured: a structured data that does not conform with the formal structure of data models associated with relational databases or other forms of data tables, but contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data
  - Ex. markup languages such as XML (Extensible Markup Language)
  - Ex. JSON (JavaScript Object Notation)

- Unstructured: data that either does not have a pre-defined data model or is not organized in a pre-defined manner
  - Ex. plain text file

Structured data: example

Field	Type	Null	Key
Host	char(60)	NO	PRI
User	char(32)	NO	PRI
Select_priv	enum('N','Y')	NO	
Insert_priv	enum('N','Y')	NO	
Update_priv	enum('N','Y')	NO	
Delete_priv	enum('N','Y')	NO	
Create_priv	enum('N','Y')	NO	
Drop_priv	enum('N','Y')	NO	
x509_issuer	blob	NO	
max_connections	int(11) unsigned	NO	
plugin	char(64)	NO	
authentication_string	text	YES	
password_last_changed	timestamp	YES	
password_lifetime	smallint(5) unsigned	YES	
account_locked	enum('N','Y')	NO	

- Thinned down table from a mysql database

Semi-structured data: delimited-separated files

- File format for storing tabular data in a plain text
- Each line of the file is a data record
- Each record consists of one or more fields, separated by a delimiter
- Common formats:
  - csv: delimiter is a comma (`,`)
  - tsv: delimiter is a tab (`\t`)
- csv file format is not standardized

tsv example: Yago knowledge base

```
<id_rB6isMnplh_H?S_LT?Qo1Fzc!>  <Jesús_Rivera_Sánchez>  <isLeaderOf>    <Pueblo_of_Naranjito>
<id_BOK!FvTDPu_H?S_1NVSTKkFbS>  <Elizabeth_II>  <isLeaderOf>    <Royal_Numismatic_Society>
<id_Uy5EwU3nX1_H?S_otuJrkvKs1>  <Richard_Stallman>      <isLeaderOf>    <Free_Software_Foundation>
<id_A?rIHtKpyX_H?S_Ap3TBzfE6b>  <Keith_Peterson>        <isLeaderOf>    <Cambridge_Bay>
<id_vzhzgmCR5Y_H?S_9xW07qYiaH>  <William_H._Seward_Jr.> <isLeaderOf>    <9th_New_York_Heavy_Artillery_Regiment>
<id_GqUh9jFAN?_H?S_A39fu5FWu4>  <Andranik>      <isLeaderOf>    <Armenian_fedayi>
<id_s60Psk1DHb_H?S_OACCn8W8Kv>  <Ramasamy_Palanisamy>   <isLeaderOf>    <Democratic_Action_Party_(Malaysia)>
<id_pII60Mnz8o_H?S_8mvRWxKXDG>  <Matt_Bevin>    <isLeaderOf>    <Kentucky_Air_National_Guard>
...
```

JSON: JavaScript Object Notation

- Open-standard format of human-readable text to transmit data objects
  - Commonly used for asynchronous browser-server communication
  - Derived from JavaScript (early 2000s)
  - Many programming languages include code to generate and parse JSON files

- Language-independent data format
  - Two competing specifications (RFC 8259 and ECMA-404)

- Objects consist of attribute-value pairs and array data types

- Basic data types: number (signed decimal), string, boolean, array, object (an unordered collection of name (key)-value pairs), null

JSON example: Yelp reviews


```
{
"business_id":"AGN788ObhwXu7rb8MEejIA",
"name":"Sheik Shoes",
"neighborhood":"Westside",
"address":"4300 Meadows Ln",
"city":"Las Vegas",
"state":"NV",
"postal_code":"89107",
"latitude":36.172259,
"longitude":-115.1963237,
"stars":1.5,
"review_count":4,
"is_open":1,
"attributes":["BusinessAcceptsCreditCards: True","RestaurantsPriceRange2: 2","WheelchairAccessible: True"],
"categories":["Shoe Stores","Fashion","Shopping"],
"hours":null,
"type":"business"
}
```

XML: Extensible Markup Language

- Markup language: defines a set of rules for encoding documents in a format that is both human and machine readable
  - Defined by an open standard W3C's XML 1.0 and other related specs
- Markup and content: text that defines the structure and content
- Tag: a markup construct that begins with `<` and ends with `>`
  - Types: start, end, empty element `<line-break />`
- Element: a logical component that either begins with a start tag and ends with a matching end tag, or consists only of an empty element tag
- Attribute: a name-value pair within a start tag or an empty element tag
- XML declaration: XML documents may begin with an XML declaration
  - Ex. `<?xml version="1.0" encoding="UTF-8"?>`

XML example

```
<person>
  <firstname>Milan</firstname>
  <lastname>Vojnovic</lastname>
  <address>
    <streetaddress>70 Girton Road</streetaddress>
    <city>Cambridge</city>
    <postcode>CB30LN</postcode>
  </address>
  <phonenumber>
    <type>home</type>
    <number>xxx</number>
  </phonenumber>
  <phonenumber>
    <type>mobile</type>
    <number>xxx</number>
  </phonenumber>
</person>
```

XML example (cont'd)

```
<person firstname="Milan" lastname="Vojnovic">
        <address streetaddress="70 Girton Road" city="Cambridge" postcode="CB30LN"/>
        <phonenumber type="home" number="xxx"/>
        <phonenumber type="mobile" number="yyy"/>
</person>
```

- firstname = "Milan" is an attribute

dblp example

```
<dblp>
<article mdate="2017-05-28" key="journals/acta/Saxena96">
<author>Sanjeev Saxena</author>
<title>Parallel Integer Sorting and Simulation Amongst CRCW Models.</title>
<pages>607-619</pages>
<year>1996</year>
<volume>33</volume>
<journal>Acta Inf.</journal>
<number>7</number>
<url>db/journals/acta/acta33.html#Saxena96</url>
<ee>https://doi.org/10.1007/BF03036466</ee>
</article>

    ...
</dblp>
```

- dblp: computer science bibliography http://dblp.uni-trier.de/
- dblp dump http://dblp.uni-trier.de/xml/

RDF: Resource Description Framework

- Family of W3C specs designed as a metadata data model expressed in XML
- Key idea: making statements about resources in expressions as triples

- Query and inference languages: SPARQL

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>
.
.
.
</rdf:RDF>
```

To probe further:
W3C semantic web standards https://www.w3.org/RDF/
W3schools https://www.w3schools.com/xml/xml_rdf.asp

Relational database management systems

- Relational database: a database based on the relational model of data
  - Proposed by E. F. Codd in 1970
  - Ex. Oracle database, IBM DB2, Microsoft SQL Server, Postgres, …
- Relations: data presented in a tabular form as a collection of tables
  - Each table consisting of a set of rows and columns
- Relational operators: for manipulating the data in tabular form
- Transactions: support for online transaction processing (OTLP)
  - Transactions: single logical operation on data
  - Atomic operations: reliable processing of database transactions (ACID)
- Programming: imperative (standard programming) and declarative (SQL queries)

ACID properties

- Atomic: when a statement is executed, every update within the transaction must succeed in order to be called successful (all-or-nothing)
- Consistent: data moves from one correct state to another correct state, with no possibility that readers could view different values that don't make sense together
- Isolated: transactions executed concurrently do not get entangled with each other
- Durable: once a transaction has succeeded, the changes will not be lost

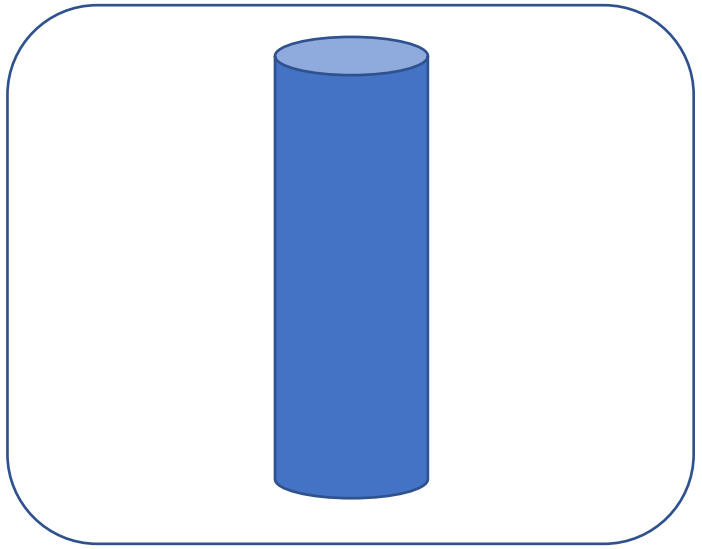
- Ex 1: if a transaction attempts to delete a customer and her order history, it cannot leave the order rows that reference the deleted customer's primary key; this is an inconsistent state that would cause errors if someone tries to read those order records
- Ex 2: if two different transactions attempt to modify the same data at the same time, then one of them will have to wait for the other to complete

- Scalability issue: may have poor availability, may use horizontal scaling but this requires distributed transactions (difficult)

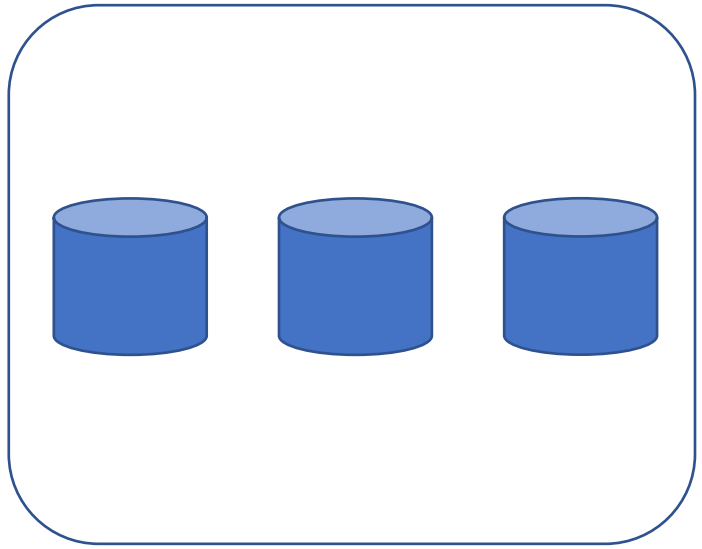
Horizontal vs vertical scaling of computer systems

- Horizontal: adding more nodes in a distributed system
  - Ex. Increasing the number of compute nodes in a cluster
- Vertical: adding resources to a single node system
  - Ex. Adding CPU or memory to a single machine

Original system



Vertical scaling
scale up or scale down



Horizontal scaling
Scale out or scale in

Data centers

- Clusters of many commodity machines (100s of thousand)
- Run by companies such as Amazon, Google, Microsoft, …

Computer system bottlenecks

- CPU utilization: when the processor is busy it cannot respond to requests for time
- Memory utilization: system does not have sufficient or fast enough RAM
  - Reduces the speed at which RAM can serve information to CPU
  - CPU starts offloading storage to significantly slower HDD or SDD memory
- Network utilization: communication between processes residing on different machines connected by a network lack the necessary communication bandwidth
- Software limitations: inefficient software implementation
- Disk: often the slowest component inside a computer

State of the art high-performance laptops

A 2017 spec:

CPU Intel i7 3.30 GHz, **6** cores, **12** threads
RAM **16GB**
HDD **1TB**

A 2020 spec:

CPU Intel i9 2.4 GHz, **8** cores
RAM **32GB**
SSD **2TB**

State of the art dev servers

- High-performance (virtual) machines hosted either in the company's data center or offered by a public cloud provider

- Some numbers:

RAM	**57GB**	**114GB**	**144GB**
Disk	**400GB** SSD	**700GB** SSD	**2TB** HDD

To probe further: GCP compute instances

https://cloud.google.com/compute/docs/machine-types

Disk HDD vs SSD

- HDD = Hard Drive [Device | Disk]
  - An electromechanical storage device with a physical spinning disk and movable read-write heads
  - Slower access time and larger latency than for SSDs

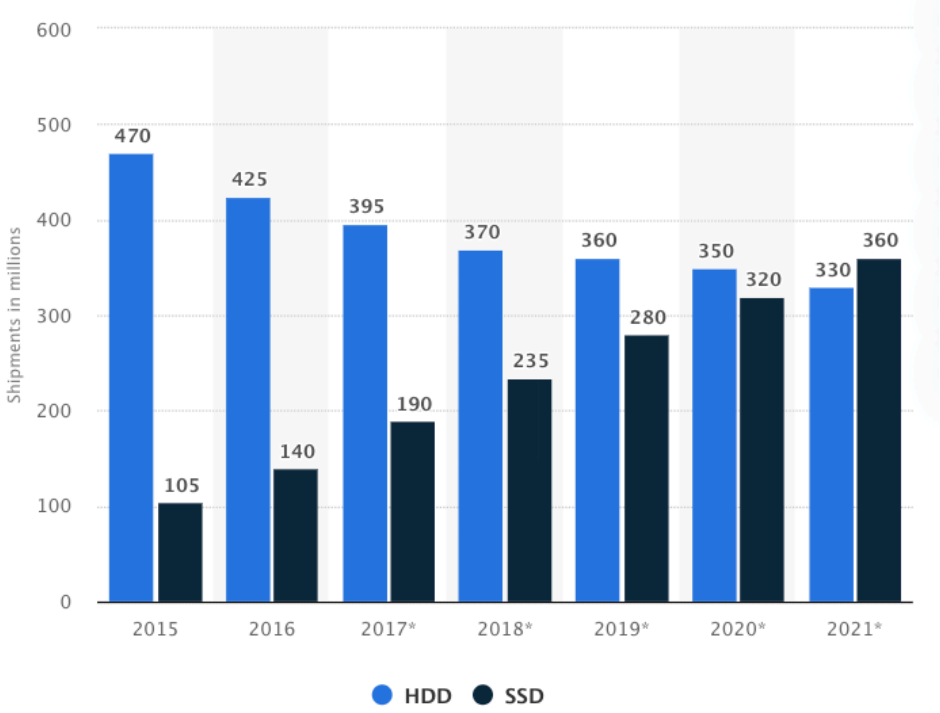
- SSD = Solid-State Drive
  - A solid-state storage device that uses integrated circuits assemblies to store data persistently
  - Typically uses flash memory (non-volatile computer memory storage medium)
  - Have quicker access time and lower latency than hard drives

To probe further: https://www.storagereview.com/ssd_vs_hdd

Hard disk price per GB

HDD prize decline

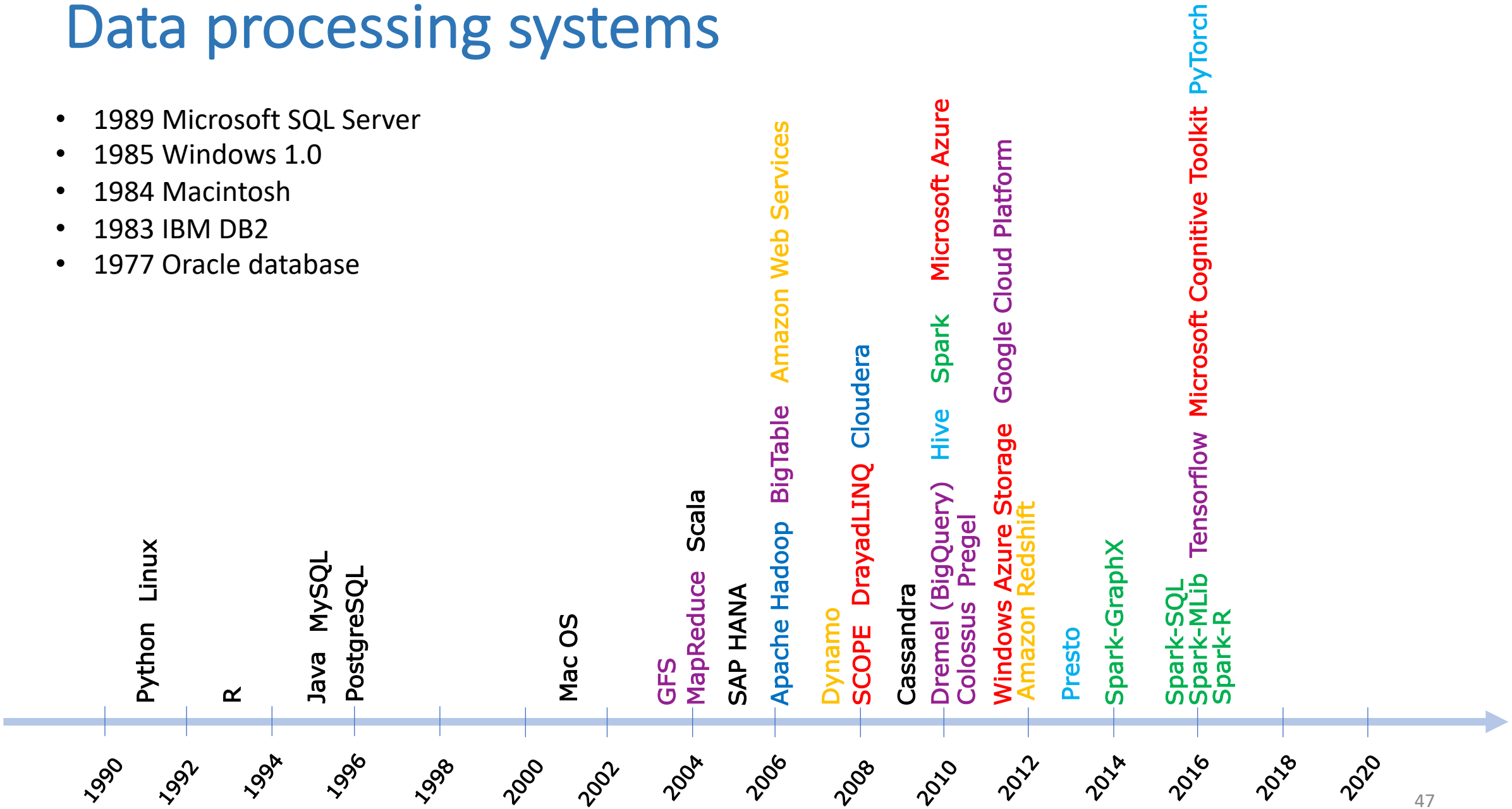
Shipments of HDD and SDD units worldwide



https://www.statista.com/statistics/285474/hdds-and-ssds-in-pcs-global-shipments-2012-2017/

Data processing systems

- 1989 Microsoft SQL Server
- 1985 Windows 1.0
- 1984 Macintosh
- 1983 IBM DB2
- 1977 Oracle database



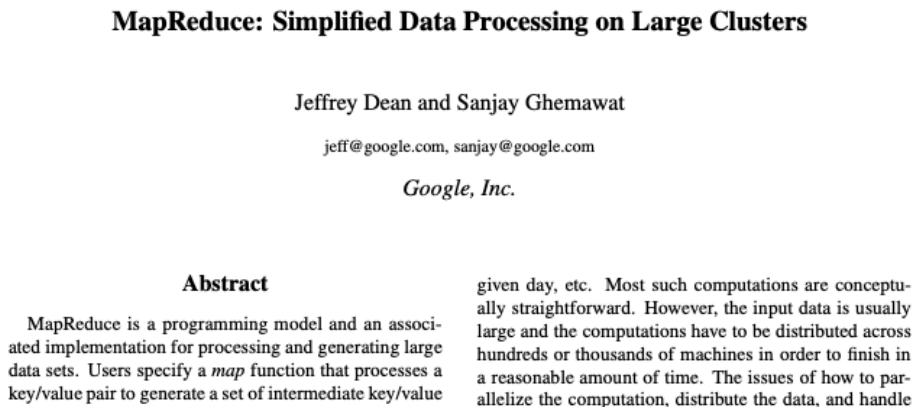
Data processing types

- Batch processing: execution of a series of jobs in a program on a computing system without manual intervention
  - Non-interactive
  - Running a mapreduce data analytics job
- Interactive computing: software which accepts input from humans as it runs
  - Running ad-hoc queries on a dataset
- Stream processing: given a sequence (stream) of data, a series of operations is applied to each element in the stream
  - Referred to as dataflow programming, event stream processing, reactive programming
- Lambda architecture: data-processing designed to handle massive quantities of data by taking advance of both batch and stream processing methods
  - Balancing latency, throughput, and fault-tolerance by using batch processing to provide comprehensive and accurate views of batch data
  - Simultaneously using real-time stream processing to provide views of online data

Examples of batch data processing tasks

- Distributed grep: `line of text if matches a user provided pattern`
- Count URL access frequency: `(url, total count)`
- Reverse web-link graph: `(target_url, list(src_url linking to target_url))`
- Term-vector per host: `(host name, list(term))`
- Inverted index: `(word, list(document id))`
- Distributed sort: `(key, value)` data by `key`

- Source: Mapreduce, OSDI 2004



MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle

Distributed file systems and mapreduce

- Distributed file systems [covered in week 2]
  - Needed new distributed file systems that can scale to large volumes of data
  - Designed for fault tolerance, for large clusters of commodity machines
  - Examples:
    - Google File System (now Colossus)
    - Apache HDFS

- Mapreduce [covered in week 3]
  - Needed a new computation / programming model
  - Required to scale, have a simple API, and cover a set of computation tasks

Apache Hadoop

- Open-source software for reliable, scalable, distributed computing
- Distributed storage and processing of large datasets using mapreduce programming model
- Computing clusters built from commodity hardware
- All modules designed with the fundamental assumption that hardware failures are common and should be automatically handled by the framework

Apache Hadoop: modules

- Hadoop Distributed File System (HDFS): storage of data on commodity machines, high aggregate bandwidth across the cluster
- Hadoop MapReduce: an implementation of the MapReduce programming model for large-scale data processing
- Hadoop YARN: a platform responsible for managing computing resources in clusters and using them for scheduling user applications
- Hadoop Common: libraries and utilities needed by other Hadoop modules

Need for new databases

- Amazon shopping cart example:
  - Need to maintain information about customer's shopping cart
  - Strict requirements for performance, reliability and efficiency

- Reliability: important as even a slightest outage has significant financial consequences and impacts customer trust
- Scalability: platform needs to be highly scalable to support continuous growth

- In the past used an Oracle database with clustering and replication
  - Reached limits to sustain the availability, scalability and performance
  - Developed Amazon Dynamo distributed key-value store
  - Traded strict consistency guarantees for availability and scalability

W. Vogels, A Decade of Dynamo http://www.allthingsdistributed.com/2017/10/a-decade-of-dynamo.html Blog, All Things Distributed

NoSQL: non-relational databases

- NoSQL began gaining popularity around 2009

- Key-value stores:
  - Data items are keys that have a set of attributes
  - All data relevant to a key is stored with the key
  - Data is frequently duplicated (unlike to SQL databases)
  - Ex: Amazon's Dynamo, caching technologies such as MemcacheD

- Column stores:
  - Referred also as wide-column stores
  - Ex: Google's Bigtable, inspired design of other systems such as Apache Cassandra

- Document stores:
  - Basic unit of storage is a document
  - Document formats: JSON, XML, YAML, …
  - Ex: MongoDB, Apache CouchDB, …

NoSQL cont'd

- Graph databases:
  - Data represented as a graph: a set of nodes and a set of edges that connect nodes
  - Both nodes and edges may have attributes
  - Use cases: social networking and semantic web applications
  - Ex. FlockDB, Neo4J, Polyglot, Amazon's Neptun

- Object databases:
  - Basic unit is an object, not relations or columns
  - Ex. db4o, InterSystems Cache

- XML databases:
  - Special form of document databases, optimized specifically for working with XML
  - Ex. Tamino and eXist

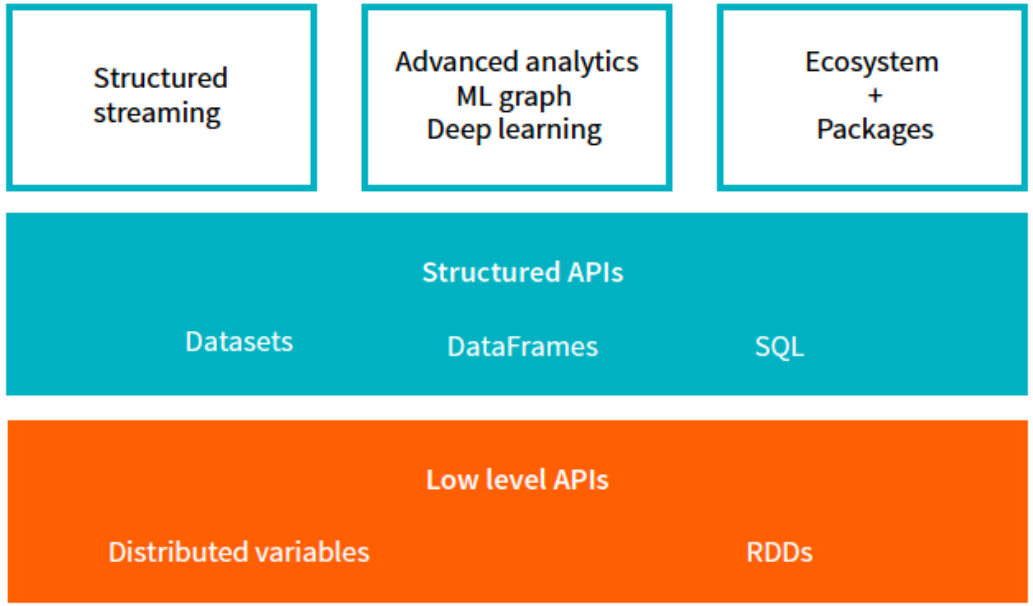
- To probe further:
  - http://nosql-database.org for a comprehensive list of NoSQL databases
  - https://db-engines.com/en/ranking for a popularity ranking of databases

Structured queries

- Execute queries using a SQL like language
- Need to allow for loading data into a structured data model
- Need to allow for fast evaluation of queries
- Want to use a SQL like API, not some imperative programming (ex. Mapreduce)
- Note: covered in week 4 (we will use Hive and Spark SQL)

Apache Spark

- Key new concept: resilient distributed data set (RDDs)
  - Designed for fault-tolerant, distributed, in-memory computing
- Integration of different types of data processing into a single system
- API interfaces in different programming languages (Scala, Java, Python, R)



Graph queries

- Graph data model: entities related with relations
  - Many data naturally fits in this model

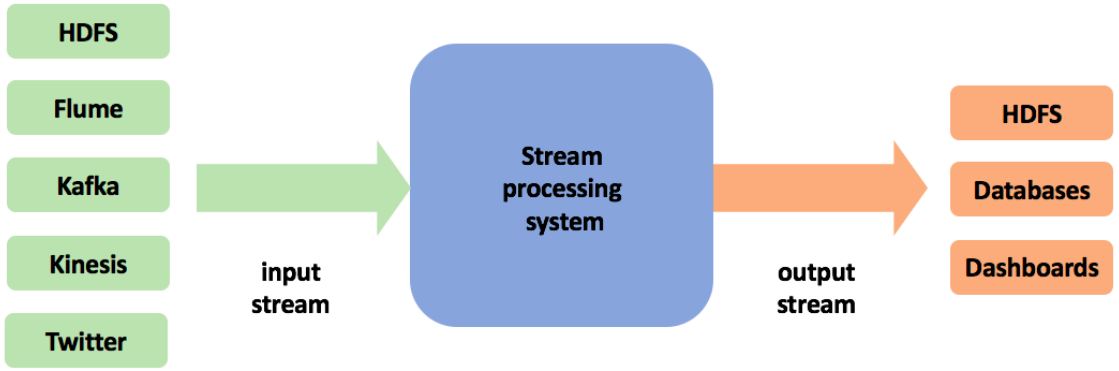
- Both entities and relations may have attributes
  - Ex. Yago: `<Richard_Stallman> <isLeaderOf> <Free_Software_Foundation>`
  - Entities: `<Richard_Stallman>` and `<Free_Software_Foundation>`
  - Relatonship: `<isLeaderOf>` attribute specifies the type of relationship 'is leader of'

- Graph processing types:
  - Aggregations: ex. node degrees
  - Iterative computations: ex. PageRank
  - Graph traversal: ex. give me all people who are leaders of Free_Software_Foundation

- Note: covered in week 5

Stream processing

- Stream processing: a programming paradigm that allows applications to more easily exploit a limited form of parallel processing



- Referred also as dataflow programming, event stream processing, and reactive programming

- Requirements: scaling to support data velocity, low latency, efficient recovery from failures, integration of batch and interactive processing

- Note: covered in week 6

Machine learning

- Training large scale machine learning models
  - Distributed iterative optimization methods for loss function minimization
  - Ex. Logistic regression, random forests, …

  [covered in week 7]

- Scalable matrix computations
  - Ex. matrix factorization for recommender systems
  - Ex. topic modeling

  [covered in week 8]

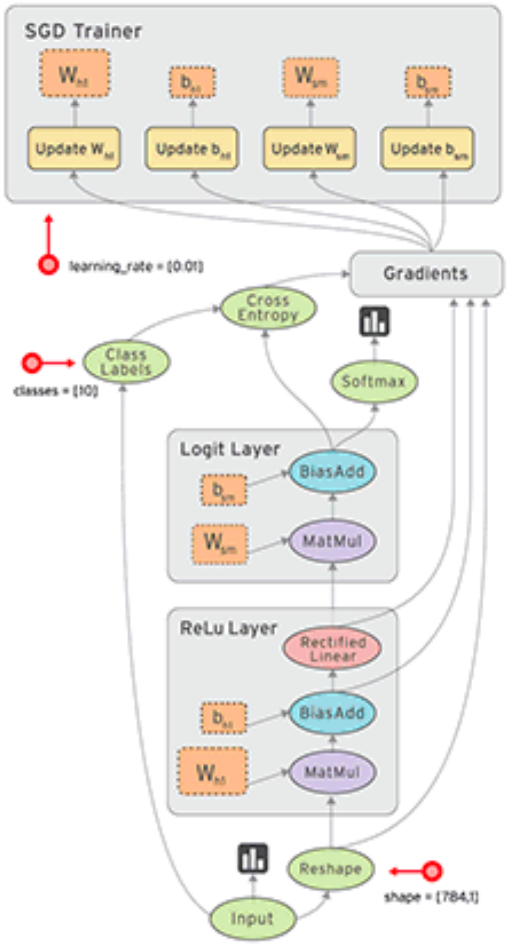
Distributed dataflow graph computations

- Dataflow programming: a programming paradigm that models a program as a directed graph of the data flowing between operations

- Key use case: learning deep neural networks

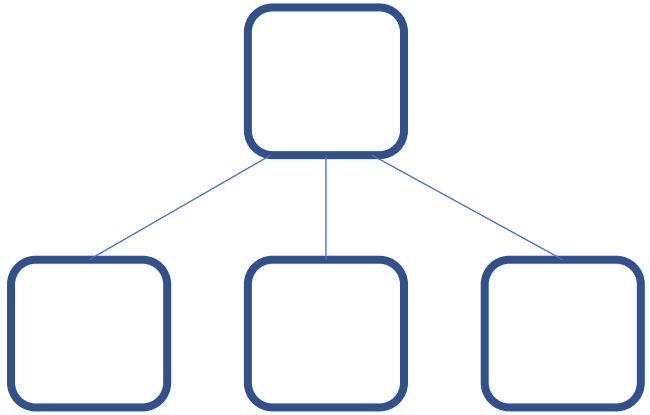
- Implemented by software frameworks such as TensorFlow

  [covered in week 10]



Seminar class 1: getting started

- Before class: get up to speed with command-line utilities
- Get started with Google Cloud Platform (GCP)
  - Running jobs on a GCP dataproc cluster
- Run a Jupyter notebook with GCP



https://github.com/lse-st446/lectures2021/blob/master/Week01/class/README.md