PDD

Lecture 1: Introduction

Prepared by Jacek Sroka

Organization

- Assessment criteria
  - Labs:
    - Working during labs/homework (20 points in total)
    - Two big programming assignments (2 * 20 points)
    - Very important to obey the deadlines
      penalties apply: -1 point per every started 12h period after the deadline (up to 14 points of penalty)
      more than one week of delay: don't bother
    - Total 60 points from labs and minimum 30 points to pass
  - Exam:
    - Written exam with total of 40 points
    - First term (only for people who pass the labs) and Second term (for everyone, grade only from exam)
    - Easy if you attended classes and participate

- There are lots of possibilities to extend the assignments (e.g., into MSc thesis)
  - We are open to good ideas (please discuss them first with your lab teacher)

    e.g. genome scale bioinformatics
  - Implementing and testing algorithms from papers
  - Working on research papers

Questions?

Materials

- Scientific papers
- Framework presentations on youtube / tutorials
- *Mining of Massive Datasets*
  Anand Rajaraman, Jure Leskovec, Jeffrey D. Ullman
- Some Coursera / edX / Stanford online courses

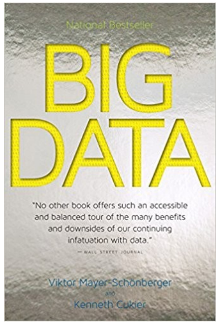
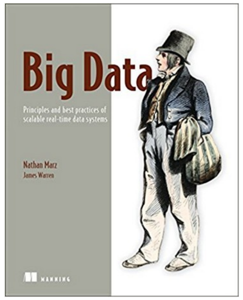
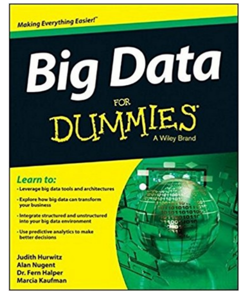
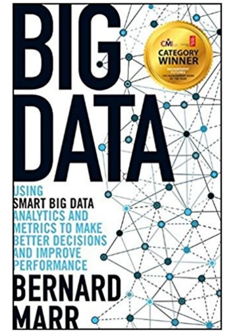
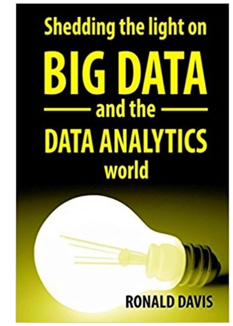
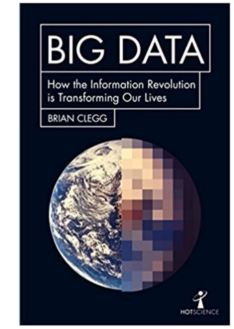
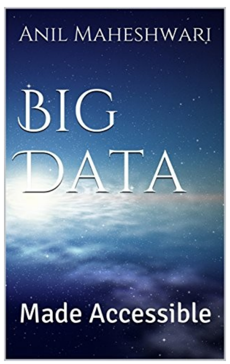
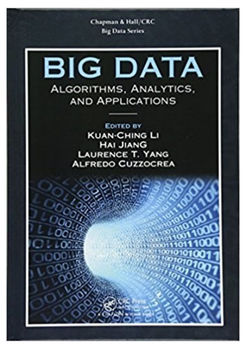
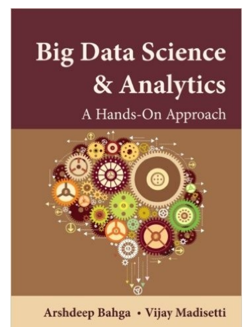
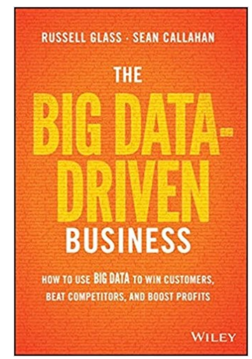
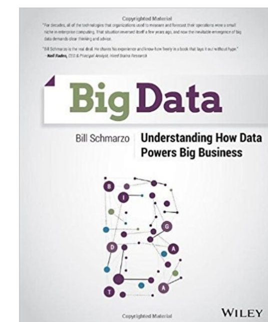
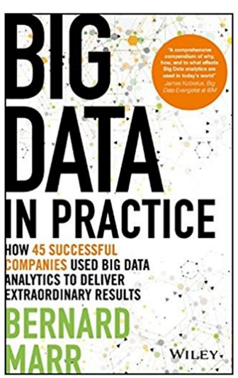
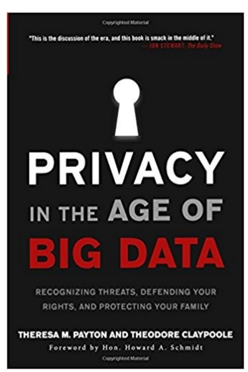
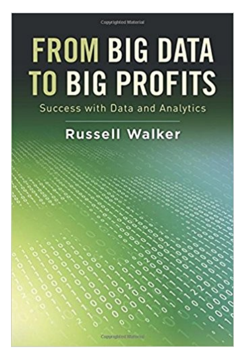
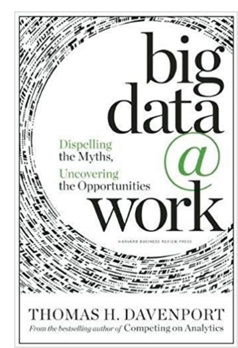
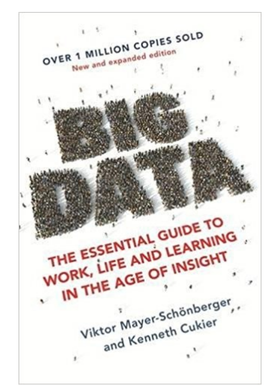
What is Big data and why it is interesting?


BIG
DATA

What is big data?

“Big data is like teenage sex: everybody talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claim they are doing it.”

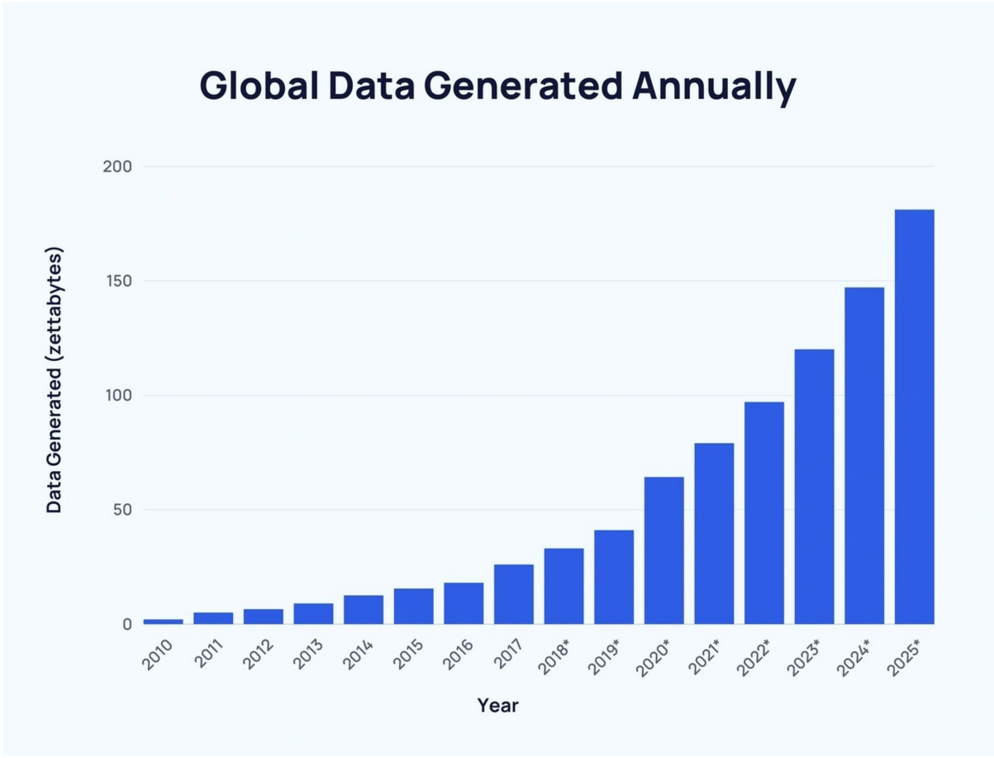
--Dan Ariely, Duke University


National Bestseller
BIG DATA
"No other book offers such an accessible and balanced tour of the many benefits and downsides of our continuing infatuation with data."
Viktor Mayer-Schönberger
Kenneth Cukier

Big Data
Principles and best practices of scalable real-time data systems
Nathan Marz
James Warren
MANNING

Making Everything Easier!
Big Data
FOR
DUMMIES
A Wiley Brand
Learn to:
Judith Hurwitz
Alan Nugent
Dr. Fern Halper
Marcia Kaufman

BIG DATA
CATEGORY WINNER
USING SMART BIG DATA ANALYTICS AND METRICS TO MAKE BETTER DECISIONS AND IMPROVE PERFORMANCE
BERNARD MARR

Shedding the light on
BIG DATA
and the
DATA ANALYTICS
world
RONALD DAVIS

BIG DATA
How the Information Revolution is Transforming Our Lives
BRIAN CLEGG

ANIL MAHESHWARI
BIG DATA
Made Accessible

Chapman & Hall/CRC
Big Data Series
BIG DATA
ALGORITHMS, ANALYTICS, AND APPLICATIONS
EDITED BY
KUAN-CHING LI
HAI JIANG
LAURENCE T. YANG
ALFREDO CUZZOCREA

Big Data Science & Analytics
A Hands-On Approach
Arshdeep Bahga • Vijay Madisetti

RUSSELL GLASS · SEAN CALLAHAN
THE
BIG DATA-DRIVEN
BUSINESS
HOW TO USE BIG DATA TO WIN CUSTOMERS, BEAT COMPETITORS, AND BOOST PROFITS
WILEY

Big Data
Bill Schmarzo
Understanding How Data Powers Big Business
WILEY

BIG DATA
IN PRACTICE
HOW 45 SUCCESSFUL COMPANIES USED BIG DATA ANALYTICS TO DELIVER EXTRAORDINARY RESULTS
BERNARD MARR
WILEY

"This is the discussion of the era, and this book is smack in the middle of it."
PRIVACY
IN THE AGE OF
BIG DATA
RECOGNIZING THREATS, DEFENDING YOUR RIGHTS, AND PROTECTING YOUR FAMILY
THERESA M. PAYTON AND THEODORE CLAYPOOLE
Foreword by Hon. Howard A. Schmidt

FROM BIG DATA
TO BIG PROFITS
Success with Data and Analytics
Russell Walker

big data @ work
Dispelling the Myths,
Uncovering the Opportunities
THOMAS H. DAVENPORT
From the bestselling author of Competing on Analytics

OVER 1 MILLION COPIES SOLD
New and expanded edition
BIG DATA
THE ESSENTIAL GUIDE TO WORK, LIFE AND LEARNING IN THE AGE OF INSIGHT
Viktor Mayer-Schönberger
and Kenneth Cukier

Wikipedia: *„Big data is an all-encompassing term for any collection of data sets so large or complex that it becomes difficult to process them using* ***traditional*** *data processing applications."*

- *Started with Internet companies Google/Facebook/Twitter/Amazon*
  (Statistics from 2012 from Wikibon Blog)
  - *In 2008, Google was processing 20,000 terabytes of data (20 petabytes) a day*
  - *Facebook stores, accesses, and analyzes 30+ Petabytes of user generated data*
  - *100 terabytes of data uploaded daily to Facebook*
  - *According to Twitter's own research in early 2012, it sees roughly 175 million tweets every day, and has more than 465 million accounts*

- *Big data was embraced by „brick and mortar"*
  - *banks, telecoms, insurance companies, ...*



Competitive advantage for brick and mortar business: Frequent itemset problem

transaction ID	items
1	{A,C,D}
2	{B,C,E}
3	{A,B,C,E}
4	{B,E}
5	{A,B,C,E}

- Analyze transaction log from store
- Goal: see what customers buy in the same transaction to organize store layout and plan promotions
  - Place snacks next to beer?
  - Offer antivirus software with new computer sales?

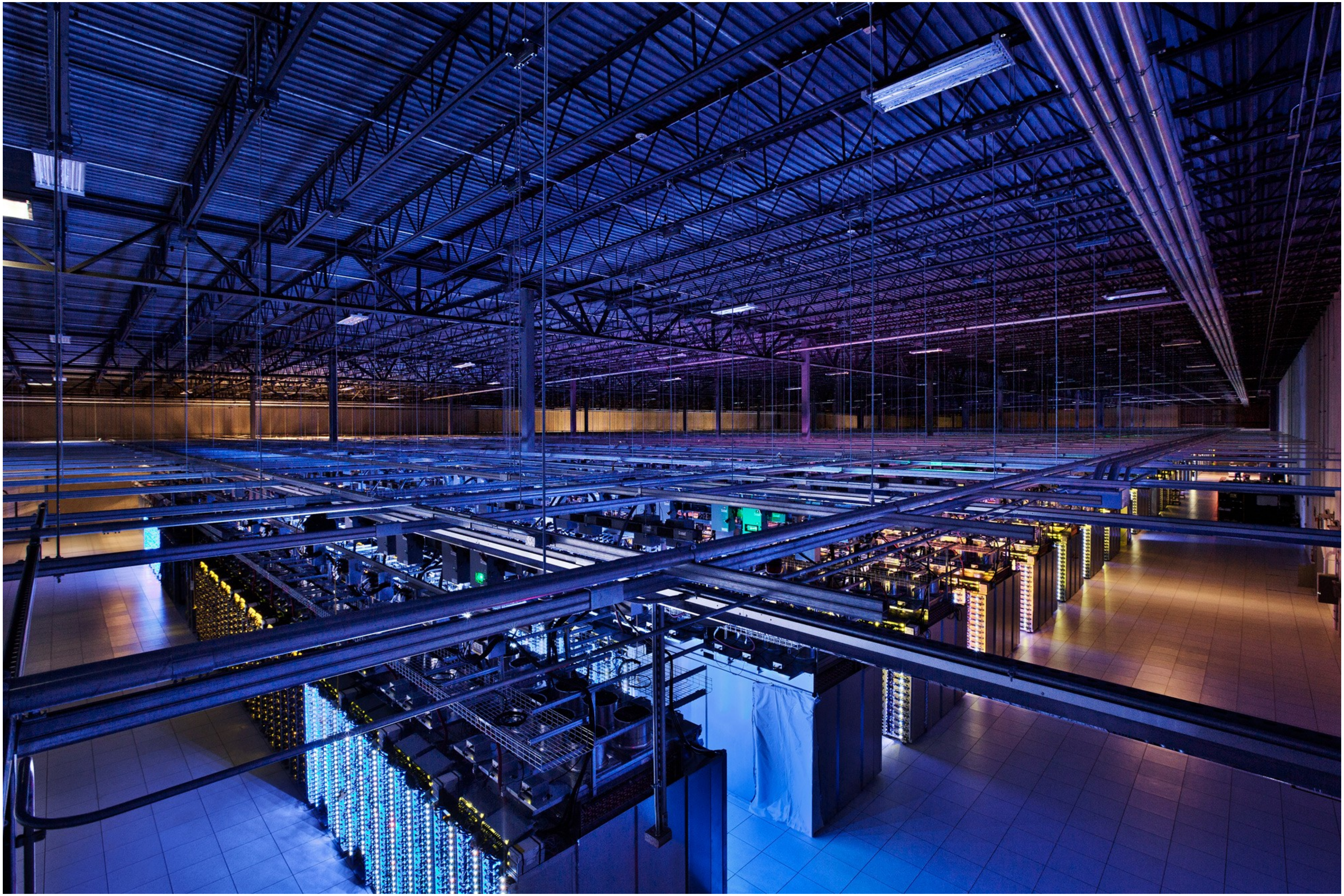
- Support: how frequently the itemset appears in the dataset support({B,E})=4/5

- Association rule learning: [‘milk]^[‘bread’]=>[‘butter’]
- Computer => antivirus_software [support=2%, confidence=60%]
  - usefulness and certainty of discovered rules
  - 2% of all the transactions under analysis show that computer and antivirus software are purchased together: Support(A -> B) = Support_count(A ∪ B)
  - 60% of the customers who purchased a computer also bought the software: Confidence(A->B) = Support(A ∪ B)/Support(A)

Competitive advantage?

What to consider
when processing Big data?

Cluster computing



Picture by: Google Inc.

Cluster computing

- Much less problems (experts do many things for us at scale and automatize)
  - Don’t need to rent server rooms, pay for electricity, hire administrators
  - No network problems (also no „Layer 0” problem – see OSI model for Layers 1-7)
  - DDoS protection and monitoring (distributed denial of service protection)
  - Updating drivers, dealing with machine failures
    (e.g. HDD breaks once in every 4 years, btw. its is more reliable when it is hot)
  - Note that other people may share the same computers!
- Examples
  - Miss Poland beauty pageant
  - Pokemon GO
  - CERN
  - Elections?
  - PRISM@NSA
- Processing data on clusters requires new frameworks
  - Take parallelism into account while designing algorithms (functional based API)
  - No need to rediscover the wheel (similar to DBMS, similar problems in many Google projects)
  - Be ready for failures (many computers, long lasting computation)
  - Deal with skew

Cluster computing

- New technologies
  - Better storage
  - Very strong computers (thousands of cores, terabytes of RAM)
- Software as a service (SaaS)
  - Even less problems, but also less control and higher price

Data science

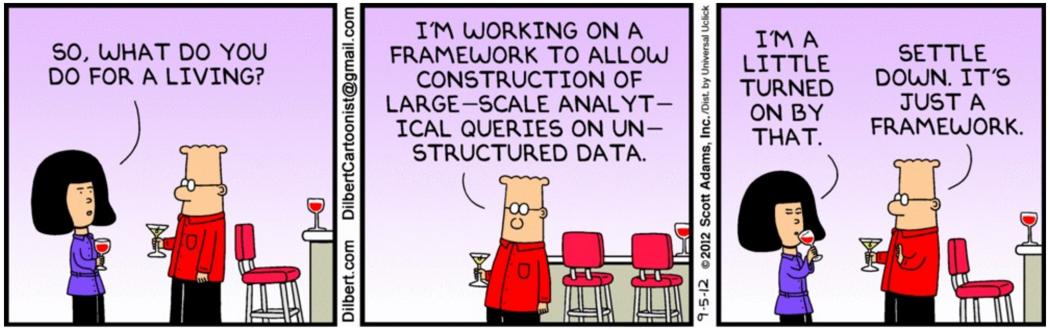
- Data science – make decisions based on how we observed the world to work
- Sometimes better to use many simple models than one sophisticated (Watson on Jeopardy; ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone)
- Often we aggregate/summarize data (pagerank, clusterization – centroids i clusteroids) to more useful representation or we find important information (frequent itemsets)
- It will get worse
  - Rate of data production is growing faster than the processing speed
  - With time there will be less and less problems for which we can use reasonable algorithms due to the growing size of input data (polynomial or even linearithmic)
  - Often result guaranteed to be good approximation is good enough
  - Different hardware is good for different problems
    - Single supercomputer with specialized memory and network
    - Cluster of commodity computers
    - Single computer with lots (TB) of RAM
      - Memory needs to be initialized
    - Graphics cards

Don’t make false conclusions

- Consider if the results makes sense and why
  - “data mining” = is the discovery of “models” for data
    where model relates to statistics, machine learning, algorithms, data bases
  - Pejorative at start (conclusion not supported by the data)
  - Bonferroni rule
    - Terrorists in hotels
      W. Bush administration's Total Information Awareness data-collection and data mining plan of 2002
    - Torture the data, and it will confess to anything
      -Ronald Coase, economist, Nobel Prize Laureate
    - The Bible Code
  - The results need interpretation (beer and diapers vs diapers and beer)
  - Correlation does not imply causation (Facebook i Princeton)
  - Watch for Weapons of Math Destruction
    - Teachers vs football
    - Hard to appeal black boxes
    - Use feedback loops
- Check for input data bias
  - Learn from ML projects mistakes
  - Failing to account for cultural bias can have consequences

General plan of the lecture

General plan of the lecture



- Technologies:
  - Spark, MLlib, Parquet
  - Stream processing (Kafka, Apache Beam)
  - ...
- Nice algorithmic ideas and how to apply them:
  - Locality Sensitive Hashing
  - Multiway joins, Minimal MapReduce algorithms
  - Distributing ML algorithms
  - ...
- What matters when distributing:
  - computation vs communication cost
  - skew
- How to pick the right tool to solve a problem
- How to avoid typical mistakes

Evolution of: Technologies, Frameworks, Tools, ...

Big data framework history

(good systems need companies that support them)

- Hadoop, Avro
  - MapReduce
  - HDFS
  - YARN from 2.0
- Spark, Parquet
  - Spark Core, SQL and Streaming
  - GraphX, GraphFrames
  - MLlib
  - AMPLab, UC Berkley, Databricks
- Google Inc.
  - Apache Beam
  - BigQuery
- Snowflake
  - Snowflake
- Nvidia
  - DASK, RAPIDS

Before Big data

- Data warehouses: storing and querying historical business data
  - 1970’s: Relational databases (w/SQL)
  - 1980’s: (more data requires multiple computers) Parallel database systems based on “shared-nothing” architectures (Gamma, GRACE, Teradata)
    - “The term shared nothing architecture was coined by Michael Stonebraker (1986) to describe a multiprocessor database management system in which neither memory nor disk storage is shared among the processors.”
    - “Each node is independent and self-sufficient, and there is no single point of contention across the system”
    - Hash partitioning
  - 2000’s: Netezza, Aster Data, DATAllegro, Greenplum, Vertica, ParAccel,...
    - 100M $ to 1B acquisitions
- On-line transaction processing (OLTP)
  - Order entry, retail sales, and financial transaction systems, …
  - Produces data for data warehouses
  - Shared-nothing also useful for scaling OLTP (1980’s: Tandem’s NonStop SQL)

Parallel database software stack

- Compiler takes query and produces query plan
- Datflow layer executes query plan
- Executed on storage managers on individual machines

- Upper layers orchestrate execution of lower layers

- Often built on top of some open source DBMS with added orchestration software: with node being instances of indexed DBMS and some

- **This stack is closed and expensive!**
- **Access data with SQL only!**

Big data in systems world

- Late 1990 – indexing and querying exploding content of the Web
  - DB technology was used but abandoned
  - Google, Yahoo! *et al* needed something else and built something
- Google
  - Google File System (GFS)
    - Files stored on 1000's of machines
    - Replication for fault-tolerance and availability
  - MapReduce (MR) programming model
    - Think about the data: user provides two simple functions (process one element + group)
    - "Parallel programming for dummies" - partitioning, message passing, fault tolerance
- Yahoo!, Facebook, and other cloned good ideas from Google's "Big Data" infrastructure
  - GFS -> Hadoop Distributed File System (HDFS)
  - MapReduce -> Hadoop MapReduce
  - Used for Web indexing, click stream analysis, log analysis, information extraction, some machine learning

MapReduce@Google

- Typical motivation: the best tools are created to avoid mundane tasks!
- Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung: **The Google file system.** SOSP 2003: 29-43
  - scalable distributed file system for large distributed data-intensive applications
  - provides fault tolerance
  - tuned to inexpensive commodity hardware
  - delivers high aggregate performance to a large number of clients
- Jeffrey Dean and Sanjay Ghemawat. 2004. **MapReduce: simplified data processing on large clusters.** In Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6 (OSDI'04), Vol. 6. USENIX Association, Berkeley, CA, USA, 10-10.
  - partitioning the input data
  - scheduling the program's execution across a set of machines
  - handling machine failures, and
  - managing the required inter-machine communication

  - Simple programming model: need to provide two functions **map** i **reduce** (think about a hammer, do not try to use it for every problem)

Tailor solution to your needs

- How to choose between: snowflake, spark, dask, open source, ...
- Pipeline vs ad hoc
  - Expressive/high level/well know API (SQL, pandas, scikit-learn, etc.) vs human optimization?
  - Hashing/Indexes?
- Computer vs cluster vs cloud vs hpc vs gpu
  - Choose best hardware and framework for your needs or framework for your hardware
- Do we need huge storage (for input, intermediate data, output)
  - store data well (binary vs textual, columnar formats, push down filters)
- Do we need to take failures into account (computation, storage)
- Do we need to scale up/down (on demand)
- Good docs/tutorials and responsive community (problems start with large deployments)
- Paid support
- Integration with other tools
- Distributing is not always good
  - Try single node first (local memory, no network, no framework overhead, no accidental complexity)
  - for some problems it is not easy/possible to design algorithms that scale well
  - for other it may work worse if you do not understand what is going on well (e.g. graph algorithms, wrong configuration of memory, checkpointing, etc.)

MapReduce

MapReduce

- Problems
  - Distributed programming is difficult (it's nice to have a simple model)
  - Network transfer is the main limiting factor (for 1Gbps transfer of 10TB takes 1 day)
  - Long running computations need to be resistant to single node failures
  - Need algorithms with low complexity and small data exchanged
- MapReduce
  - Simple programming model, well tested framework
    - Lots of nice algorithmic ideas available
    - Easy to use as intermediate layer for other high level languages like SQL
    - High **accidental complexity**, but lots of projects that help with that like Crunch, Cascading or Oozie
  - Data replication (based on HDFS)
    - chunk servers are compute servers
    - When possible schedule processing to a node with the data
  - Very pessimistic approach to failures
    - Big overhead for iterative algorithms (data mining, graph processing)
- We won't program in MapReduce but we will use it as introduction to Massively Parallel Computation (MPC) model
  - Goal for the first lab solve simple problems in MapReduce and implement in Spark (which has better API but don't use it yet)

Hello World

- Goal: count number of occurrences of a given word in a large text file
  - Web sever log analysis – finding popular URLs
  - Finding stop words in a given language
- What we need to take into account
  - File is too large to fit in memory
  - Pairs `<word,count>` do not fit in memory

Hello World

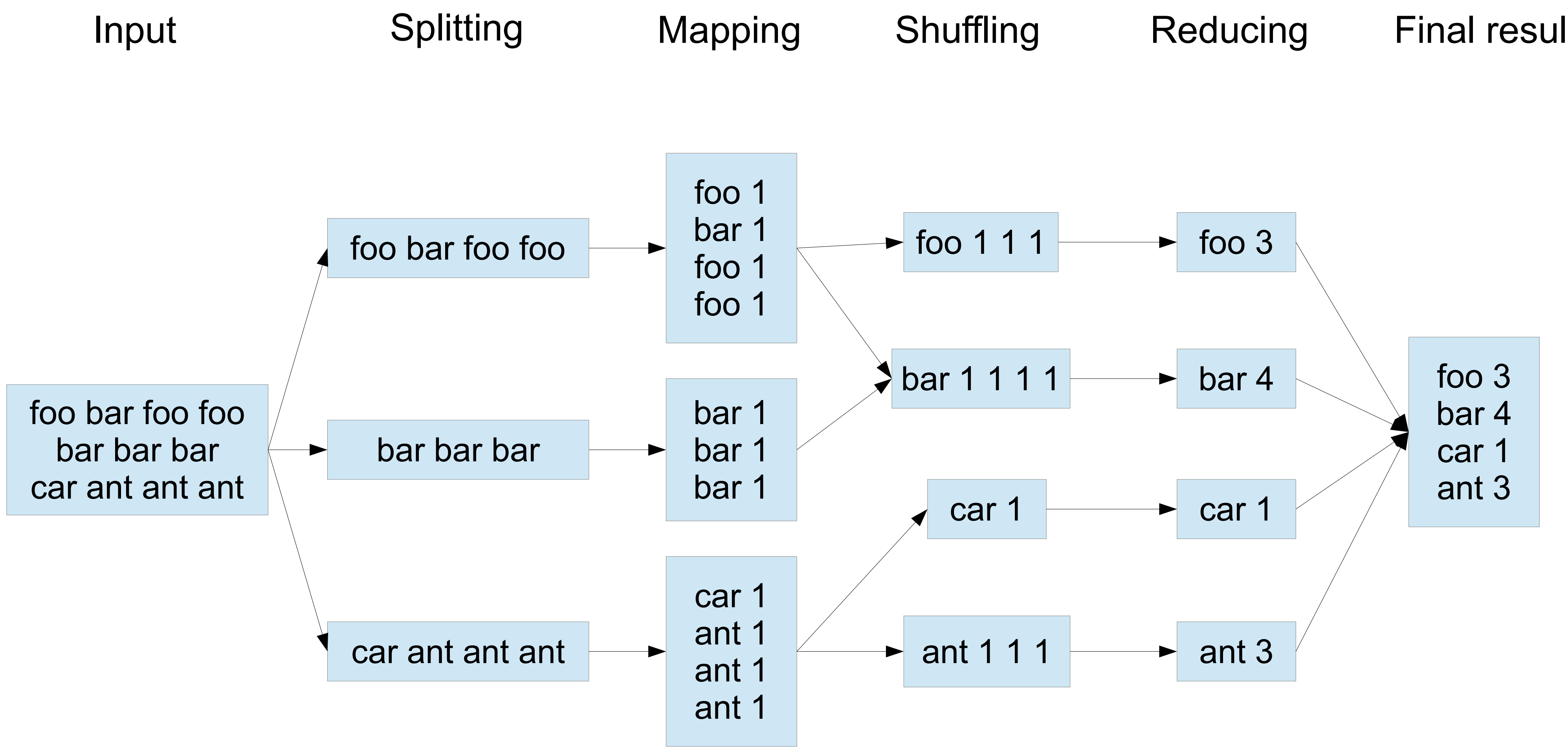
- Goal: count number of occurrences of a given word in a large text file
  - Web sever log analysis – finding popular URLs
  - Finding stop words in a given language
- What we need to take into account
  - File is too large to fit in memory
  - Pairs `<word,count>` do not fit in memory
- Ideas for a solution
  - Use HashTable
    - Easy to distribute (process independently, then integrate results)
  - Start with some grouping or sorting, e.g., with some file system tool, then process smaller files
    - Also easy to distribute

MapReduce

- Map
  - Scan input record by record, do some preprocessing
  - Choose group for the record (based on some key) write in local file
  - Easy to distribute
- Shuffle/Sort
  - Shuffle local files so that records with the same key end up at the same node
- Reduce
  - Assuming that groups can be processed on nodes, i.e., are not materialized in memory or are small enough

- Lots of reasonable ways to choose a key for grouping (depends on the use case)
  - Whole word
  - First letter
  - 0 or 1 for the first or second part of alphabet
  - The goal is to balance the groups as much as possible

MapReduce example



MapReduce c.d.

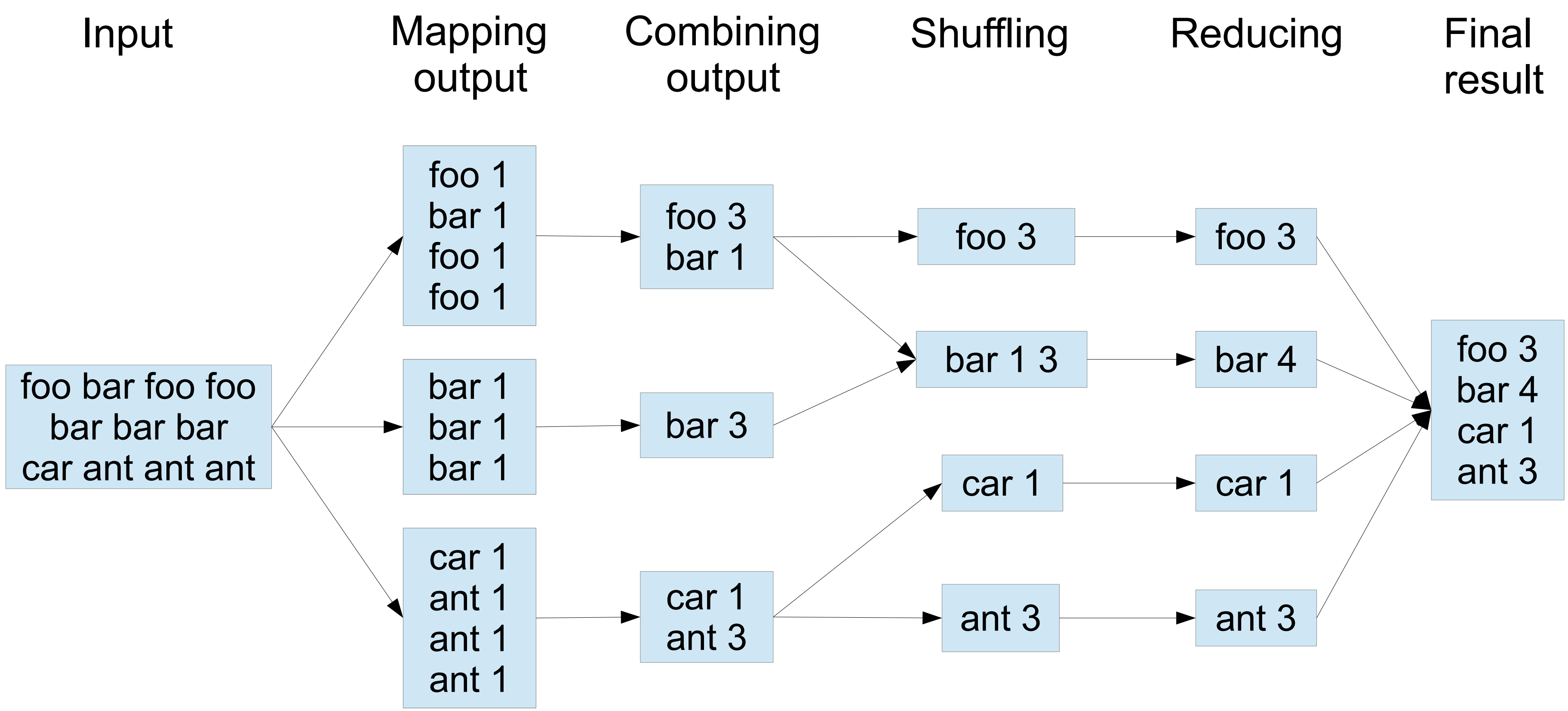
- Input: collection of key/value pairs
- Map(k,v) → <k',v'>*
  - Execute once for each input pair
  - Results in an arbitrary number of intermediate key/value pairs
- Reduce(k', (v1', v2', …)) → <k'',v''>*
  - Execute once for each k'
  - Has access to all values for that key
- Example:
  - Read input in parts, e.g., line by line
  - map: for each line generate pairs <word, 1>
  - reduce: return length of the values list
  - map and reduce are executed on multiple nodes
- Disc access is sequential not random (VERY important for HDDs, but for SDDs too)

Terminology

- Map-Reduce job =
  - Map function (inputs -> key-value pairs)
  - +
  - Reduce function (key and list of values -> outputs)
- Map and Reduce Tasks/Processes apply Map or Reduce function to (typically) many of their inputs.
  - Unit of parallelism
- **Mapper** = application of the Map function to a single input
- **Reducer** = application of the Reduce function to a single key-(list of values) pair

Combiners

- Usually we can pre-aggregate Map task ouputs with reduce function



- Works only for commutative (a + b = b + a) and associative (a + (b + c) = (a + b) + c) functions
  - e.g. for sum we can, but not for avg (1+2+3)/3 != ((1+2)/2 + 3/1)/2
  - often this can be fixes – for avg by emitting (sum, count) and using different Comb. and Red.
  - But not always – median