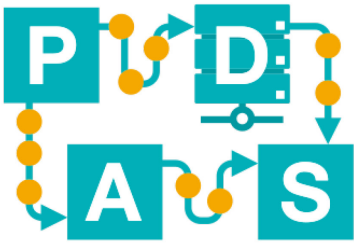
Big Data

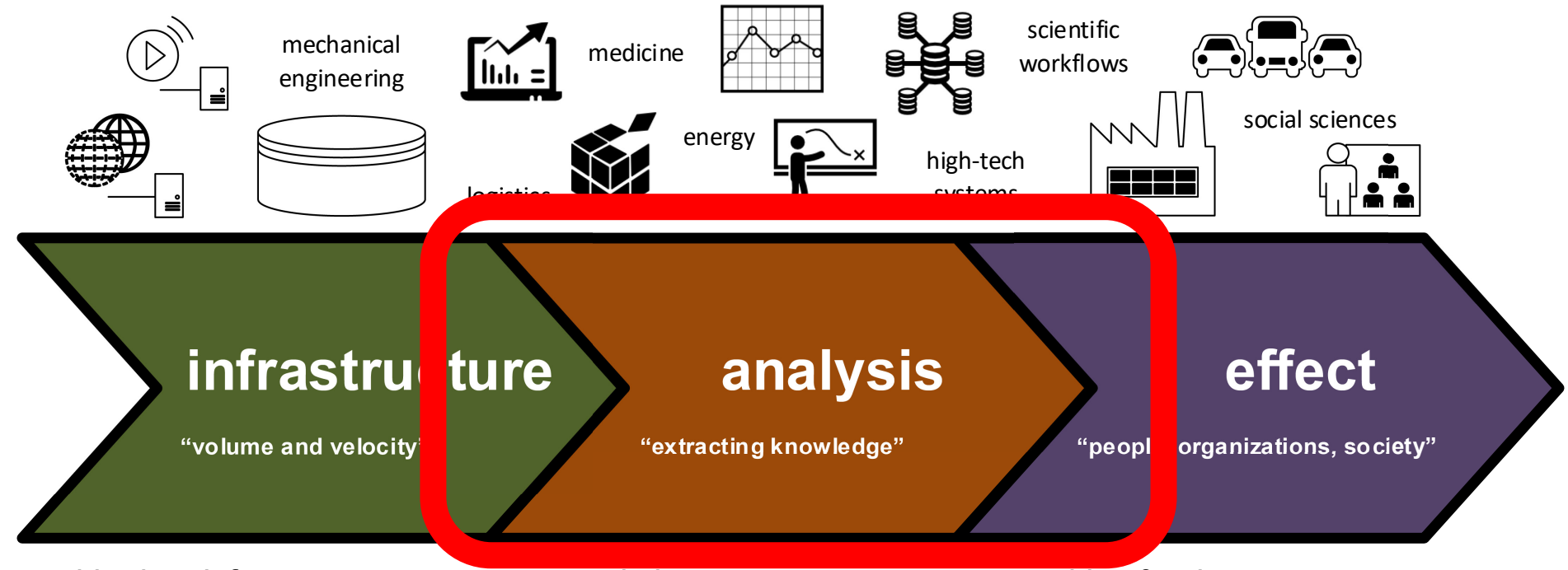
Lecture 22

IDS-L22



Chair of Process and Data Science

RWTH AACHEN UNIVERSITY



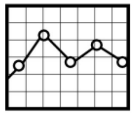
- big data infrastructures
- distributed systems
- data engineering
- programming
- security
- ...

- statistics
- data/process mining
- machine learning
- artificial intelligence
- visualization
- ...

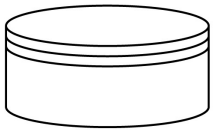
- ethics & privacy
- IT law
- operations management
- business models
- entrepreneurship
- ...

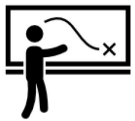
Chair of Process and Data Science


mechanical engineering

medicine


scientific workflows

logistics
energy

high-tech systems
last two lectures

infrastructure

"volume and velocity"

- big data infrastructures
- distributed systems
- data engineering
- programming
- security
- ...

analysis

"extracting knowledge"

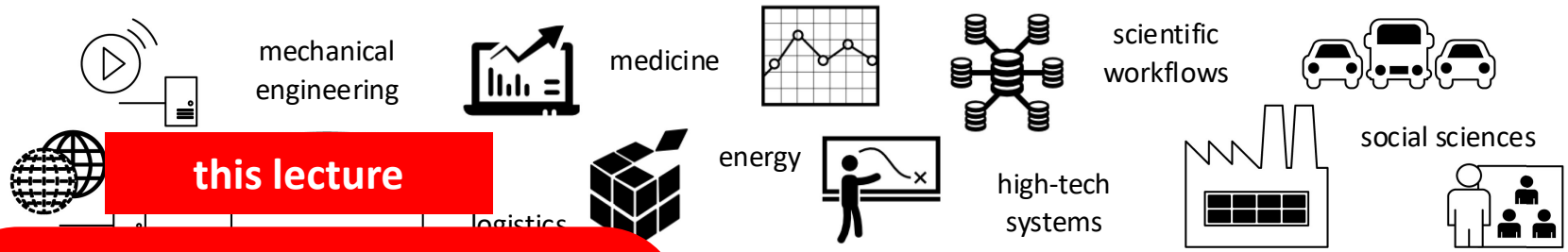
- statistics
- data/process mining
- machine learning
- artificial intelligence
- visualization
- ...

effect

"people, organizations, society"

- ethics & privacy
- IT law
- operations management
- business models
- entrepreneurship
- ...

Chair of Process and Data Science



infrastructure

"volume and velocity"

- big data infrastructures
- distributed systems
- data engineering
- programming
- security
- ...

analysis

"extracting knowledge"

- statistics
- data/process mining
- machine learning
- artificial intelligence
- visualization
- ...

effect

"people, organizations, society"

- ethics & privacy
- IT law
- operations management
- business models
- entrepreneurship
- ...

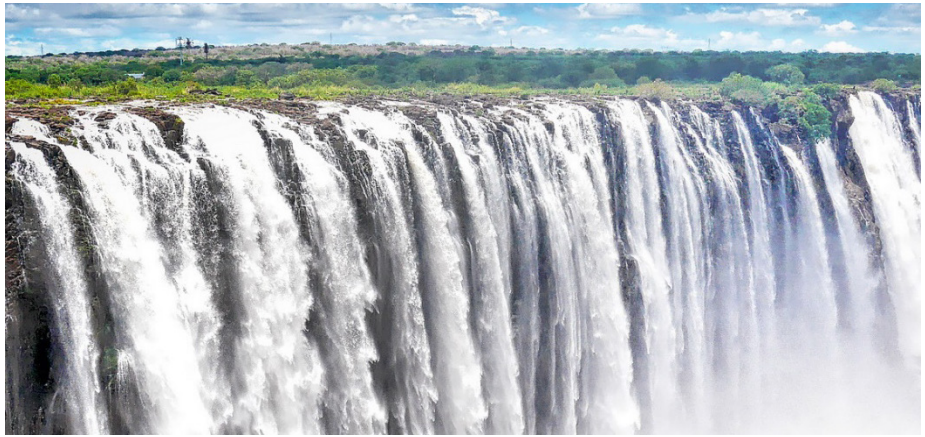
Chair of Process and Data Science

Outline of Today's Lecture

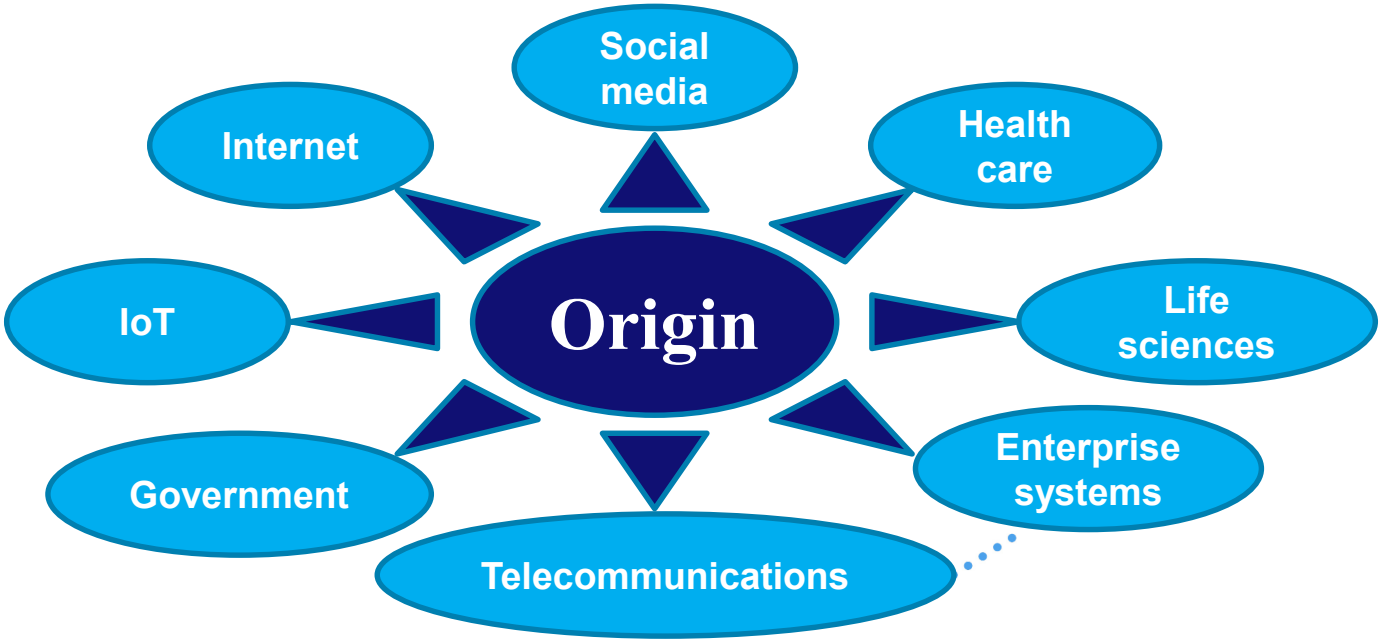
- Big Data: Relevance and Applications
- Characteristics of Big Data: The 4 V's
- Big Data Infrastructures
- MapReduce
- Streaming
- Streaming k-means

Big Data: Relevance and Applications

From Statistics to Big Data



Where does Big data come from?



Data Explosion

- Around 8.4 billion connected things are in use worldwide in 2017.
- The number will reach 20.4 billion by 2020.
- The consumer segment is the largest user of connected things with 5.2 billion units in 2017.
- Regionally, Greater China, North America and Western Europe together represent 67 percent of overall internet of things in 2017.

https://wearesocial.com/blog/2018/01/global-digital-report-2018

Applications of Big Data

- **What benefits can the massive volume of available data bring?**

Applications of Big Data

- **Employ data to predict crime hotspots**

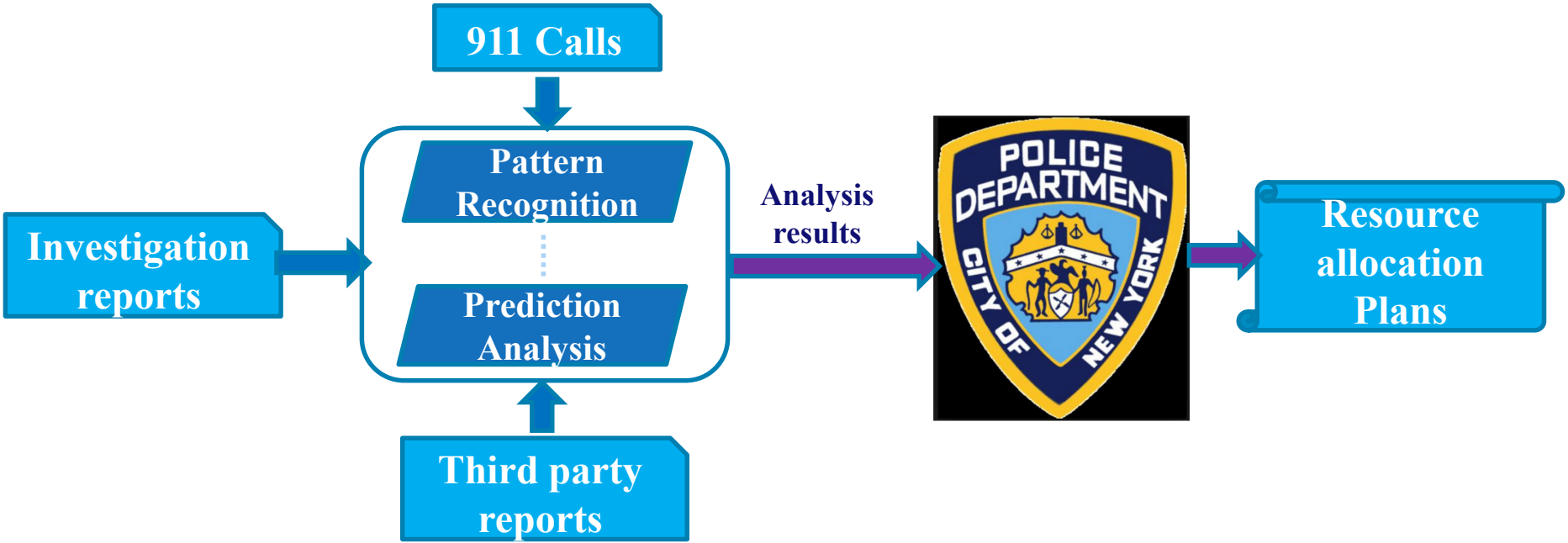
How to allocate police resources ???????

1. Budgets are limited.
2. Hard to judge where crimes happen.

http://www.dhl.com/content/dam/downloads/g0/about_us/innovation/CSI_Studie_BIG_DATA.pdf

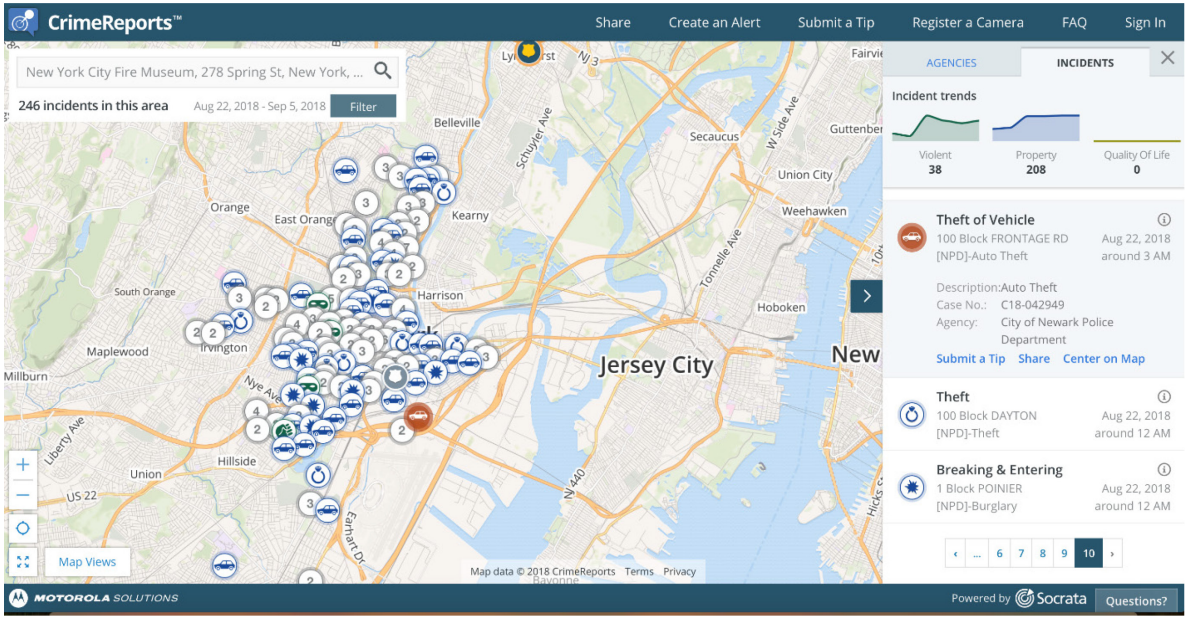
Applications of Big Data

- **Employ data to predict crime hotspots**



Applications of Big Data

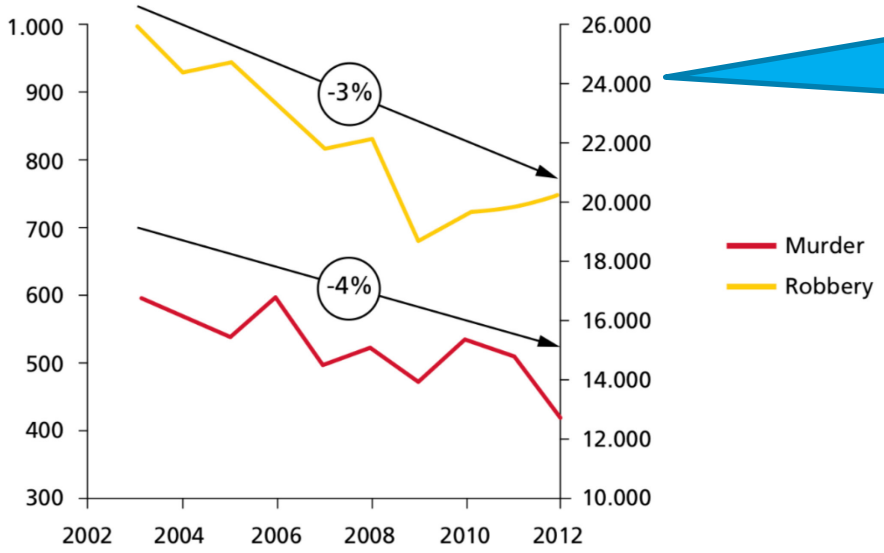
- **Employ data to predict crime hotspots**



- **Information integration.**
- **Descriptive statistics.**
- **Data mining.**
- **Visualization.**
- **Visual analytics.**

Applications of Big Data

- **Employ data to predict crime hotspots**

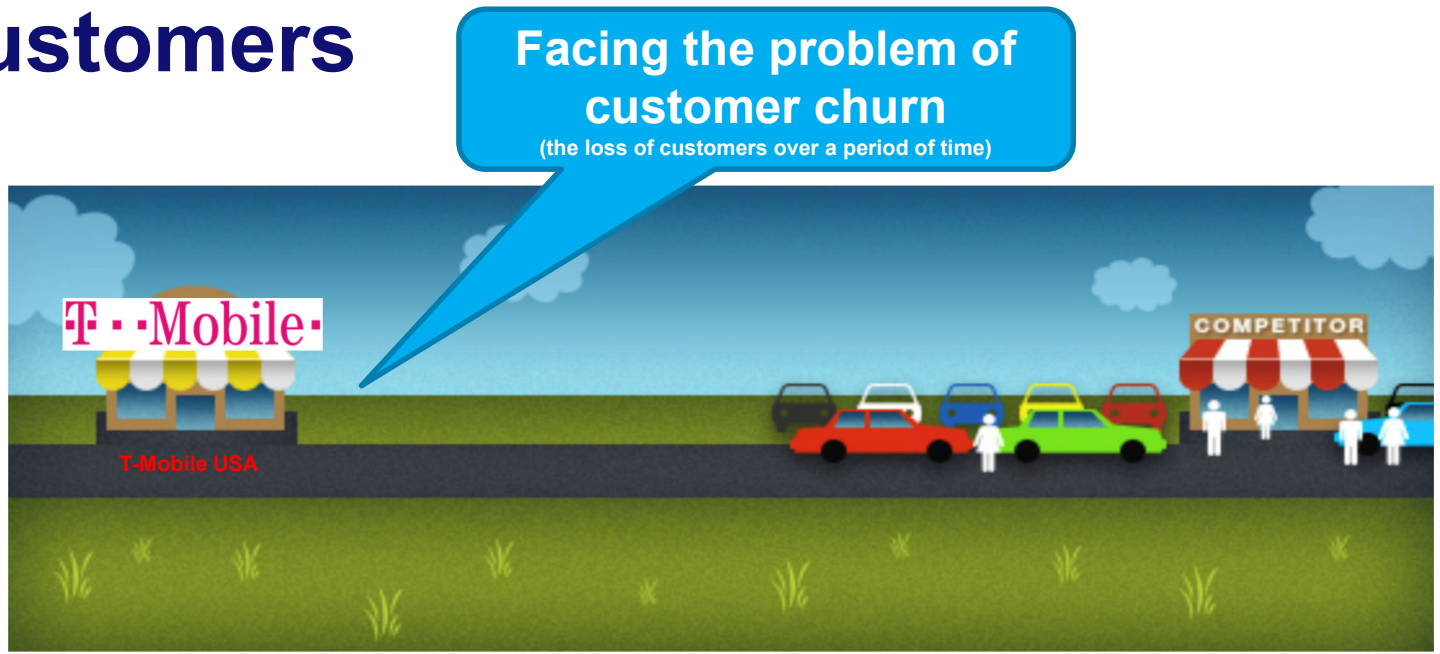


Using the correlated data sets, the NYPD is able to effectively target the deployment of manpower and resources

"By introducing statistical analysis and geographical mapping of crime spots, the NYPD has been able to create a "bigger picture" to guide resource deployment and patrol practice. Now the department can recognize crime patterns using computational analysis, and this delivers insights enabling each commanding officer to proactively identify hot spots of criminal activity."

Applications of Big Data

- **Social network analysis for retaining customers**



http://www.dhl.com/content/dam/downloads/g0/about_us/innovation/CSI_Studie_BIG_DATA.pdf

Applications of Big Data

Communication network elements

Round one petabyte of data

Historical information

Billing system

Discover tribe leaders

Maintain tribe leaders well

Search for loyal customers

Maintain loyal customers well

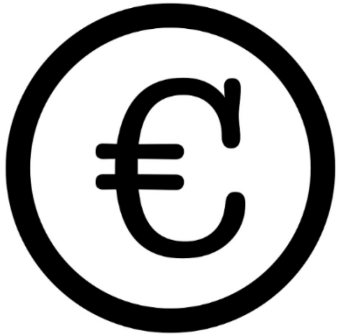
Chun rates plunged by 50%

http://www.dhl.com/content/dam/downloads/g0/about_us/innovation/CSI_Studie_BIG_DATA.pdf

Applications of Big Data

- **Data-driven marketing**

Amazon‘s database contains billions of data points from tens of millions of customers and products !

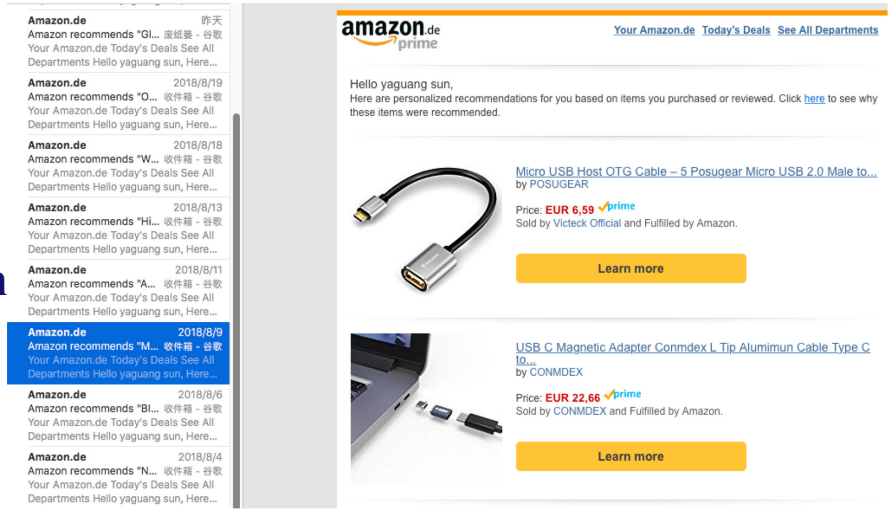


Applications of Big Data

- **Data-driven marketing**

data-driven recommender system

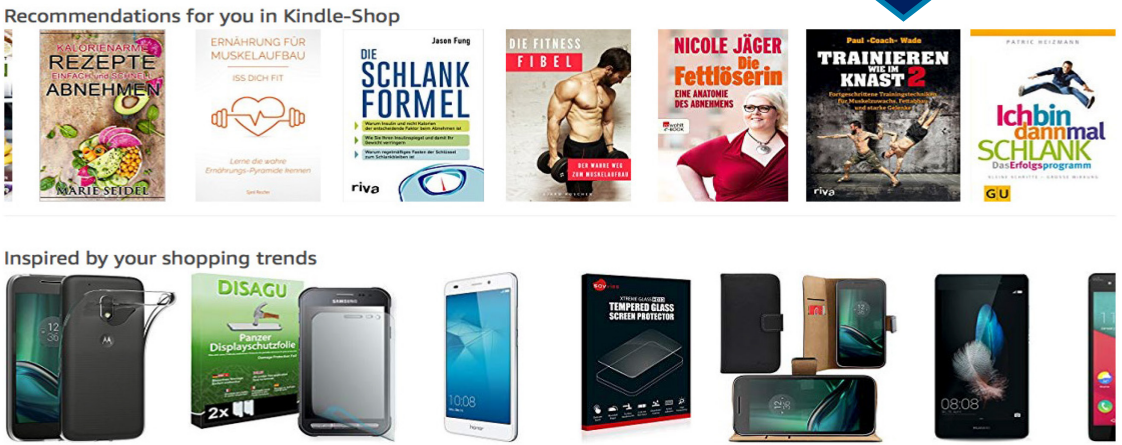
Offline recommendation



Applications of Big Data

On-site recommendation

data-driven recommender system



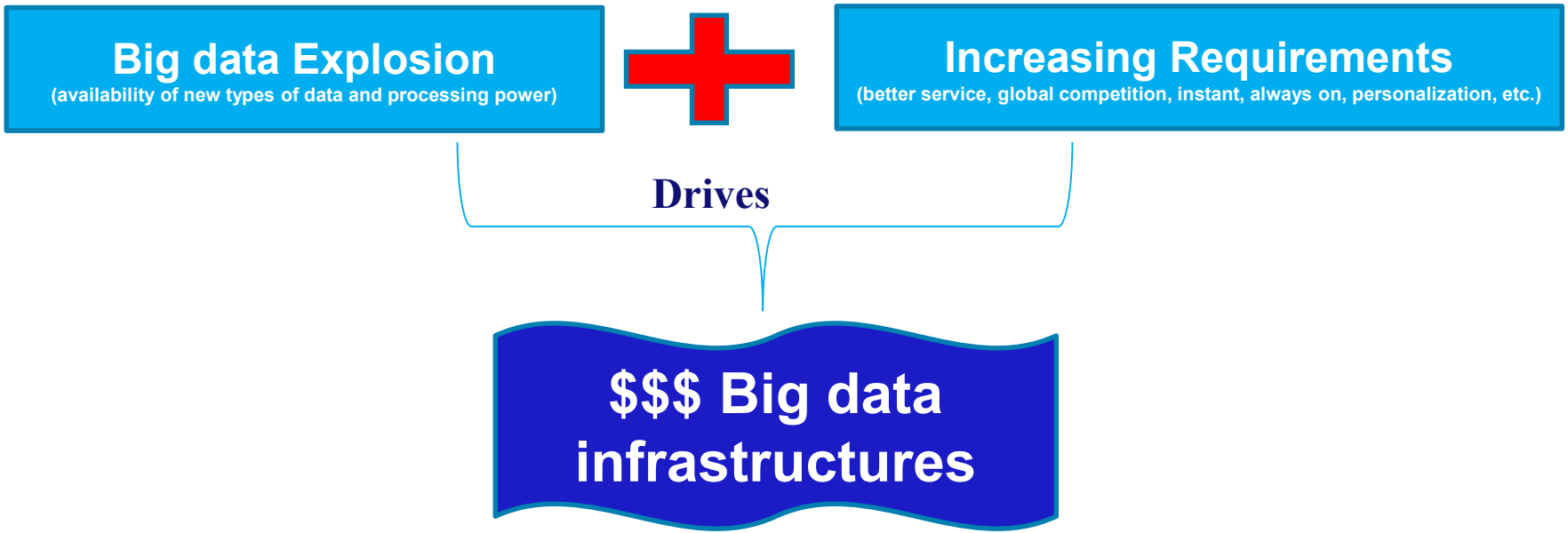
Applications of Big Data

35% of purchases from Amazon.com‘s customers are brought by its recommendation engine.

75% of of what people watch on Netflix comes from recommendations.

https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers.

What drives the development of Big Data infrastructures?



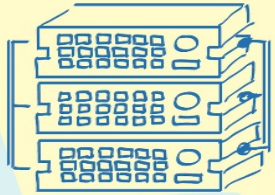
Chair of Process and Data Science

Characteristics of Big Data: The 4 V's

BIG DATA

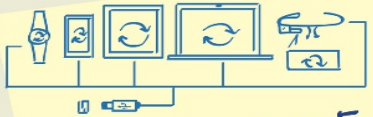
VOLUME

SCALE OF DATA



VARIETY

FORMS OF DATA



ANALYSIS OF DATA-FLOW

VELOCITY

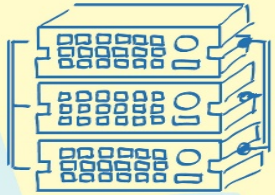
UNCERTAINTY OF DATA

VERACITY

BIG DATA

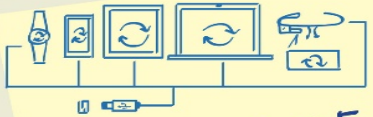
VOLUME

SCALE OF DATA



VARIETY

FORMS OF DATA



ANALYSIS OF DATA-FLOW

VELOCITY

UNCERTAINTY OF DATA

VERACITY

Characteristics of big data

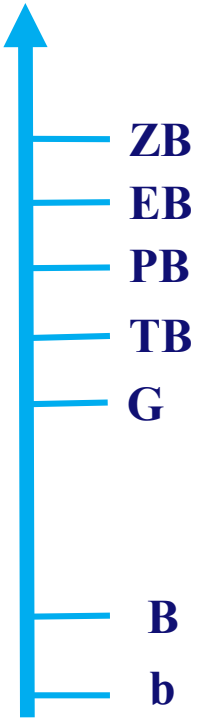
V1

- **Volume = Data Size**

1	b	bit
8 x b	B	Byte
1024 x B	KB	Kilobyte
1024^2 x B	MB	Megabyte
1024^3 x B	GB	Gigabyte
1024^4 x B	TB	Terabyte
1024^5 x B	PB	Petabyte
1024^6 x B	EB	Exabyte
1024^7 x B	ZB	Zettabyte

Characteristics of big data

V1



180 ZB ≈ estimated amount of digital data in 2025

0.5 ZB ≈ entire World Wide Web in 2009

5 EB ≈ all words ever spoken by human beings

1 PB ≈ 2.7 billion facebook images

1 TB ≈ 300 hours of high quality video

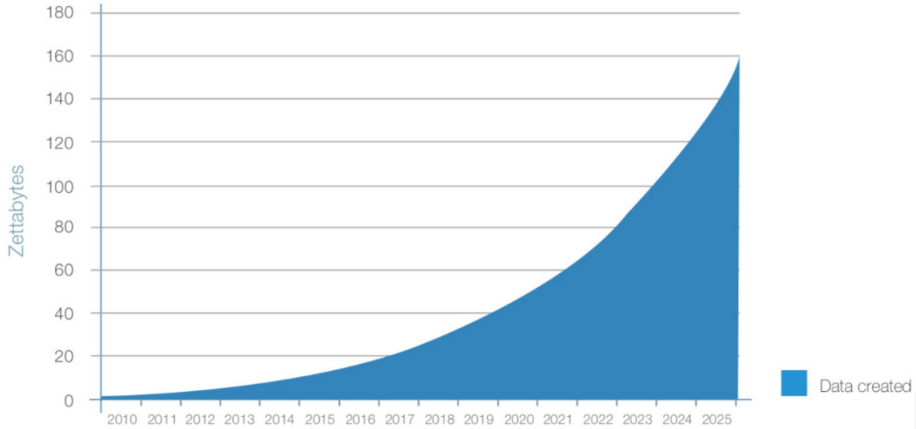
1 GB ≈ 7 minutes of HDTV video at 19.39 Mbit/s

1 Byte = one character

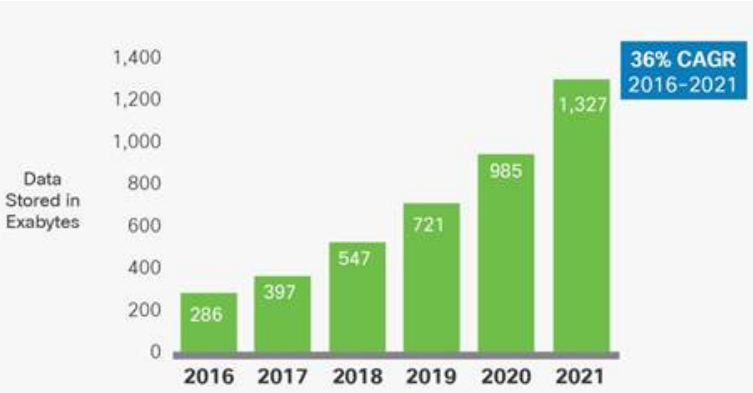
1 bit = 0/1

All point in the same direction

V1



Source: IDC's Data Age 2025 study, sponsored by Seagate, April 2017



Source: Cisco Global Cloud Index, 2016-2021.

Examples

V1

- Snapchat users share 527,760 photos every day
- 456,000 tweets are sent on Twitter every day
- We send 16 million text messages every minute
- Spotify adds 13 new songs every minute
- Uber riders take 45,788 trips every minute
- 990,000 Tinder swipes every minute
- 600 pages are edited in Wikipedia every minute
- 154,200 calls on Skype every minute
- More than 300 million photos are uploaded every day in Facebook
- Every minute 510,000 comments posted and 293,000 statuses are updated in Facebook

Key question

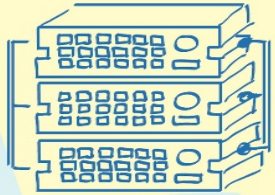
V1

How to collect, store, and process all of these data in an economical and reliable manner, instantly?

BIG DATA

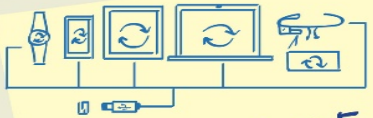
VOLUME

SCALE OF DATA



VARIETY

FORMS OF DATA



ANALYSIS OF DATA-FLOW

VELOCITY

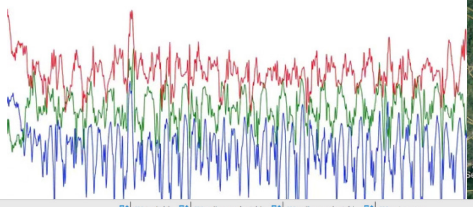
UNCERTAINTY OF DATA

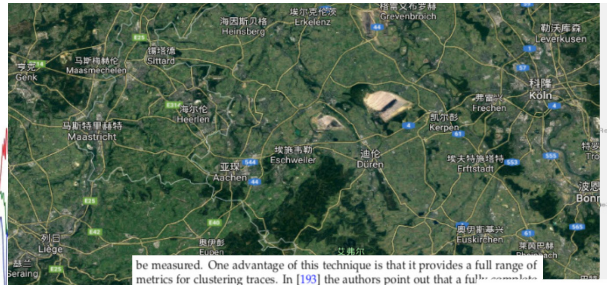
VERACITY

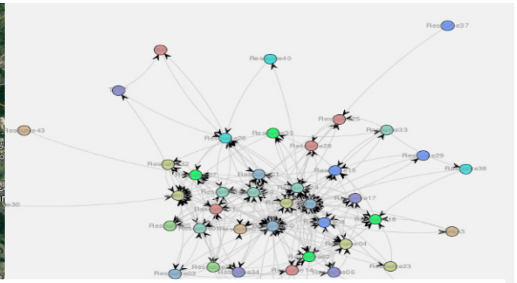
Characteristics of big data

V2

- **Data are more heterogeneous than before**







	act_id_	task_id_	call_proc_inst_id_	call_case_inst_id_	act_name_	act_type_
678	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
679	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
680	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
681	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
682	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
683	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
684	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
685	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
686	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
687	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
688	eventSchadenmeldungEingetroffen	[NULL]	[NULL]	[NULL]	Schadenmeldung eingetroffen	messageStartEvent
689	gatewayVersicherungsschutzJoin	[NULL]	[NULL]	[NULL]		exclusiveGateway
690	serviceTaskAuszahlungsbetragErmitteln	[NULL]	[NULL]	[NULL]	Auszahlungs-betrag ermitteln	serviceTask
691	ruleTaskEntscheidenObPruefbericht	[NULL]	[NULL]	[NULL]	Entscheiden ob Prüfbericht notwendig	businessRuleTask
692	gatewayPruefberichtNotwendig	[NULL]	[NULL]	[NULL]	Prüfbericht notwendig?	exclusiveGateway
693	eventSubprocessPruefberichtStart	[NULL]	[NULL]	[NULL]		startEvent
694	subprocessPruefbericht	[NULL]	[NULL]	[NULL]		subProcess
695	messageTaskPruefberichtBeauftragen	[NULL]	[NULL]	[NULL]	Prüfbericht beauftragen	sendTask
696	messagePruefberichtEingetroffen	[NULL]	[NULL]	[NULL]	Prüfbericht empfangen	receiveTask
697	gatewayVSNRJoin	[NULL]	[NULL]	[NULL]		exclusiveGateway
698	gatewayVSNRUebergeben	[NULL]	[NULL]	[NULL]	VSNR übergeben?	exclusiveGateway
699	serviceTaskVertragAnhandVSNRLesen	[NULL]	[NULL]	[NULL]	Vertrag anhand VSNR lesen	serviceTask
700	intermediateThrowEventSchadenAngelegt	[NULL]	[NULL]	[NULL]	Schaden angelegt	intermediateNoneTh
701	serviceTaskSchadenAnlegen	[NULL]	[NULL]	[NULL]	Schaden anlegen	serviceTask
702	ruleTaskLeistungsanspruchPruefen	[NULL]	[NULL]	[NULL]	Leistungs-anspruchÜberÜfen	businessRuleTask
703	intermediateThrowEventLeistungsanspruchGepr	[NULL]	[NULL]	[NULL]	Leistungs- anspruch geprÜft	intermediateNoneTh
704	gatewayBestehtVersicherungsschutz	[NULL]	[NULL]	[NULL]	Besteht ¶Versicherungs-schutz?	exclusiveGateway

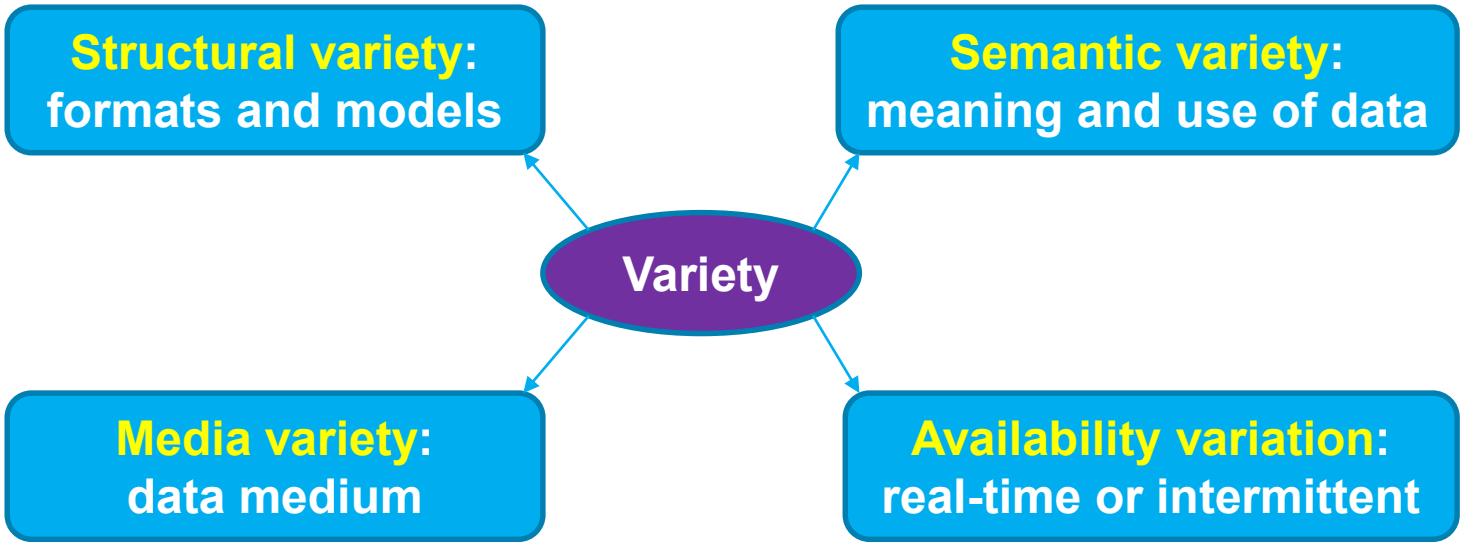
be measured. One advantage of this technique is that it provides a full range of metrics for clustering traces. In [193] the authors point out that a fu
model (with high fitness) discovered may support a high variety of
that are not registered in event log, as a result some significant struct
may be concealed in the mined model. Such a problem can be dealt
sidering the metric soundness [9] which measures the percentage o
of the mined model that are recorded in the log among all of the beh
ported by the model. An efficient technique is proposed in [193] w
the whole process into a set of distinct sub-processes based on a gre
which makes sure the further division of a process will lead to ano
ingly sound sub-process. This method can also help solve the prob
complexity of the initial model. Context-aware trace clustering te
proposed in [195] and [35]. In [195] the authors indicate that the
based on subsequences of different lengths are context-aware for the
model and can reveal some set of common functions. Two traces that
conserved features in common should be gathered in the same cluste
authors present an edit distance-based approach for distributing trac
ters such that each cluster consists of traces with similar structural
The cost of edit operations is associated with the contexts of activitie
calculated edit distance between traces is more accurate. The sequen
technique based on first-order Markov chains is presented in [196]
nique learns a potential first-order Markov model for each cluster
expectation-maximization algorithm. A sequence is assigned to a cl
is able to generate it with higher probability. The technique prop
paper also inherits the idea from sequence clustering, the difference
nique represents each cluster with a set of separate sequences (sig

The complexity of data storage and analysis is greatly increased with a variety of structured and unstructured data!!

Characteristics of big data

V2

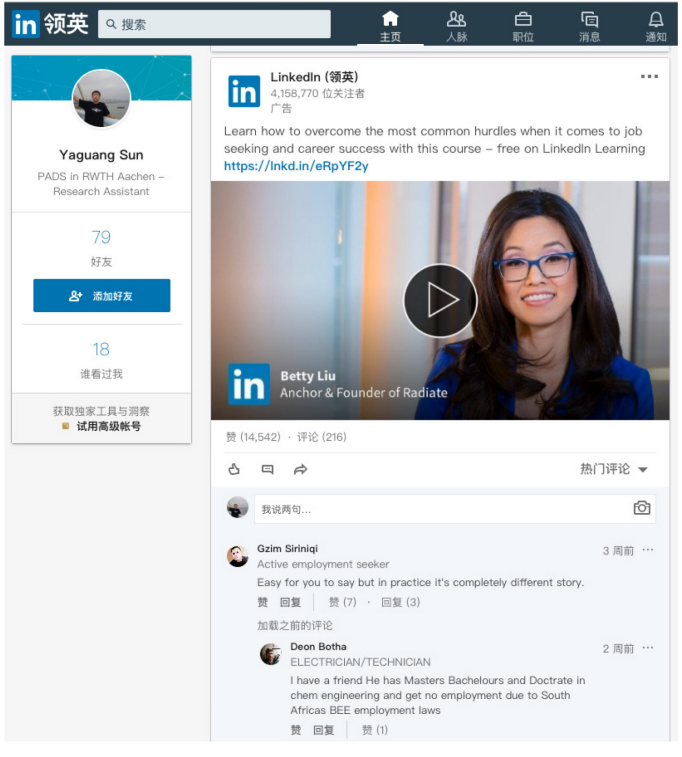
- **Four types of variety**



Characteristics of big data

V2

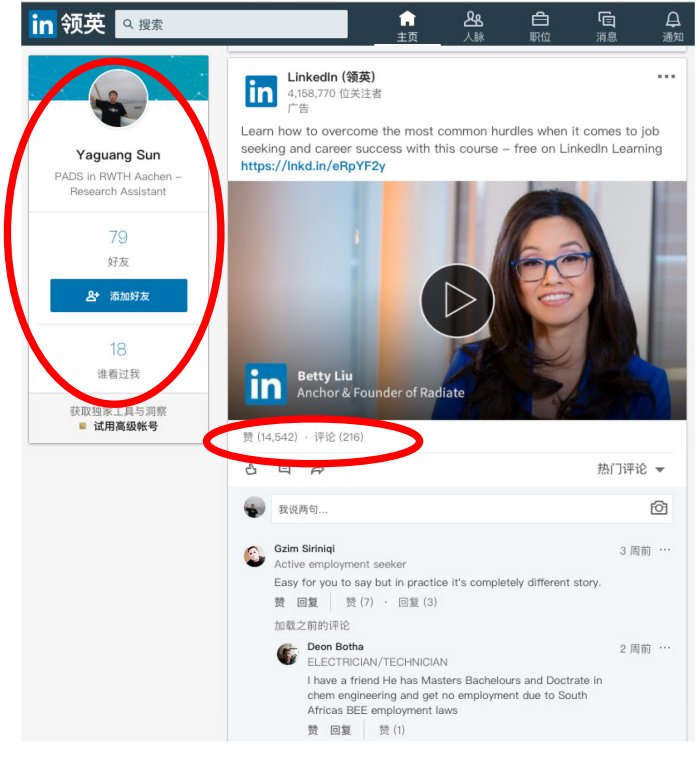
- **Variety in social network**



Characteristics of big data

V2

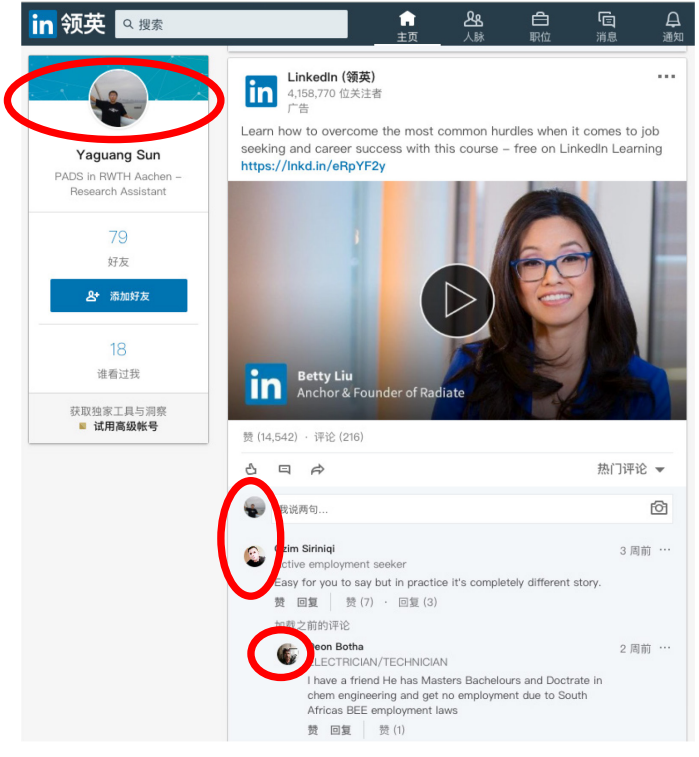
- **Variety in social network**
  - **Structured data**



Characteristics of big data

V2

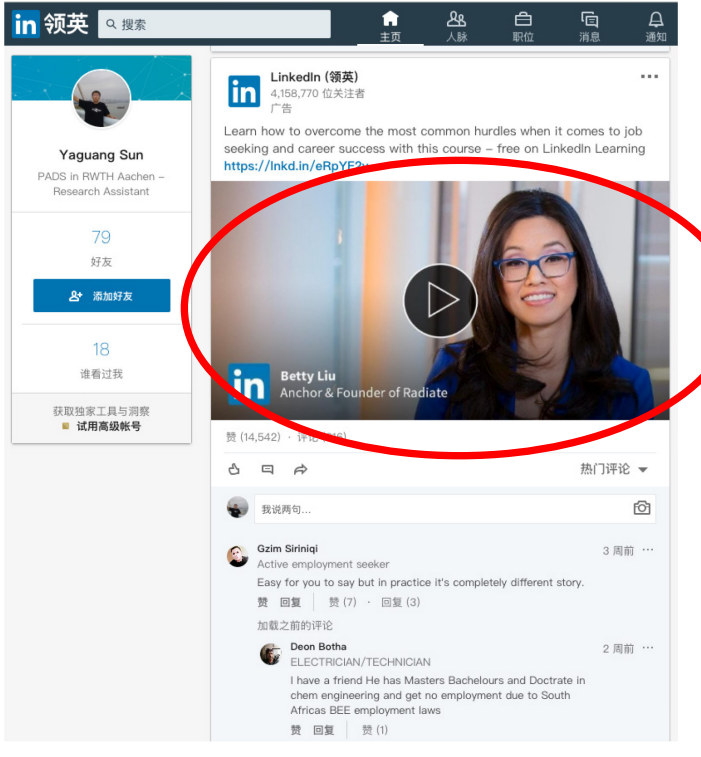
- **Variety in social network**
  - **Structured data**
  - **Pictures**



Characteristics of big data

V2

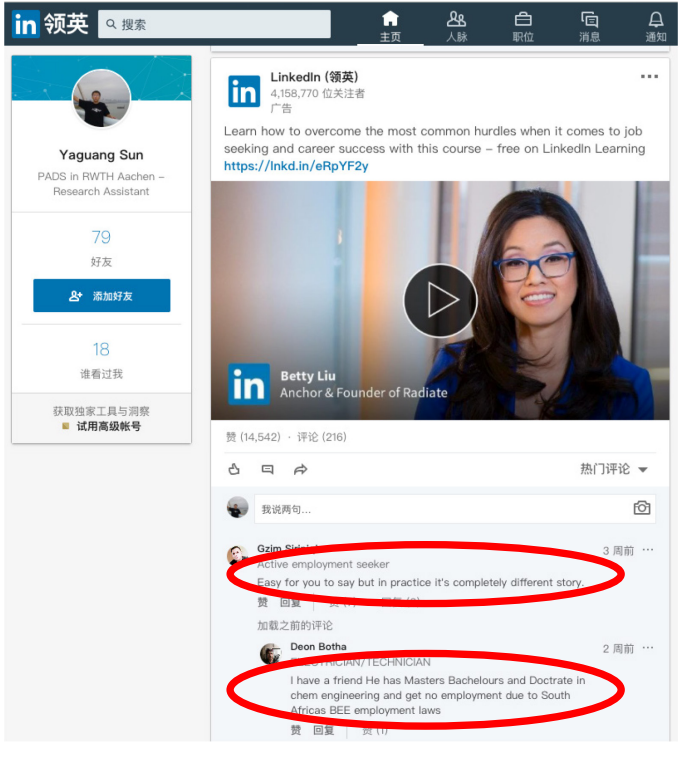
- **Variety in social network**
  - **Structured data**
  - **Pictures**
  - **Video**



Characteristics of big data

V2

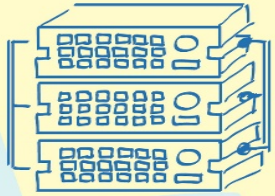
- **Variety in social network**
  - **Structured data**
  - **Pictures**
  - **Video**
  - **Unstructured text**



BIG DATA

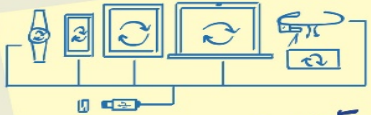
VOLUME

SCALE OF DATA



VARIETY

FORMS OF DATA



ANALYSIS OF DATA-FLOW

VELOCITY

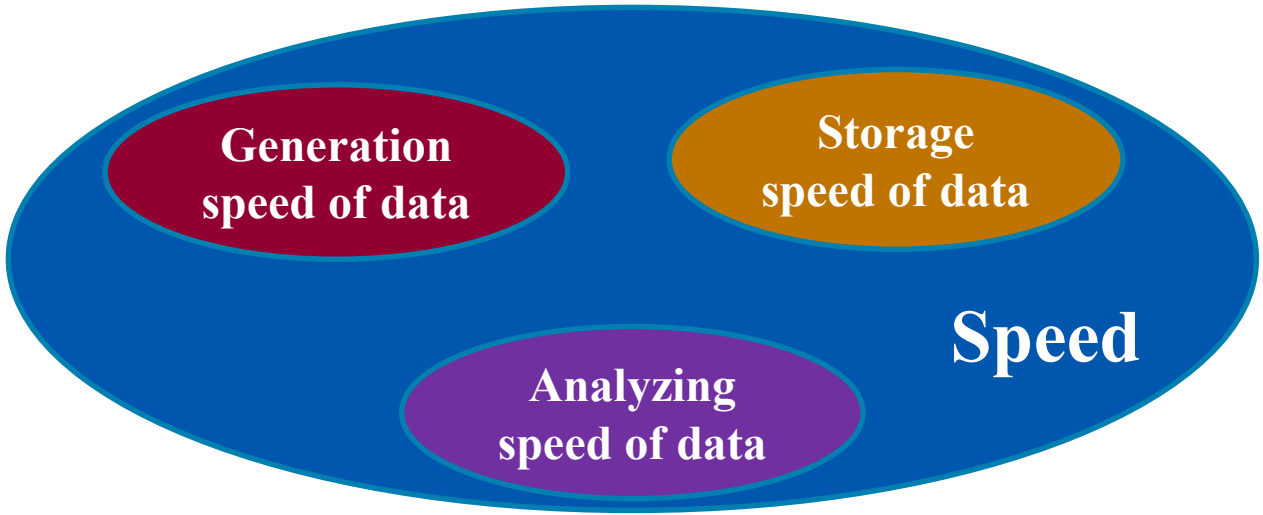
UNCERTAINTY OF DATA

VERACITY

Characteristics of big data

V3

- **Velocity = Speed**



Characteristics of big data

V3

- **Exponential growth of data**

The Patt & Whitey's GTF engine for a Bombardier C Series jetliner contains 5000 sensors, which generate 10 GB of data per second.

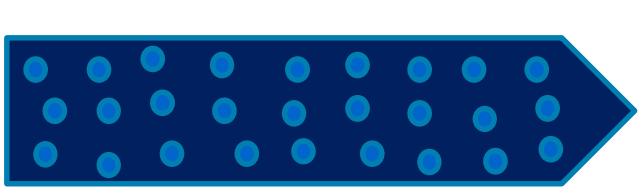
http://aviationweek.com/connected-aerospace/internet-aircraft-things-industry-set-be-transformed

Characteristics of big data

V3

- **Processing data in real-time to match its generation speed is very important in some applications**

Alibaba‘s singles day shopping festival



Data flow

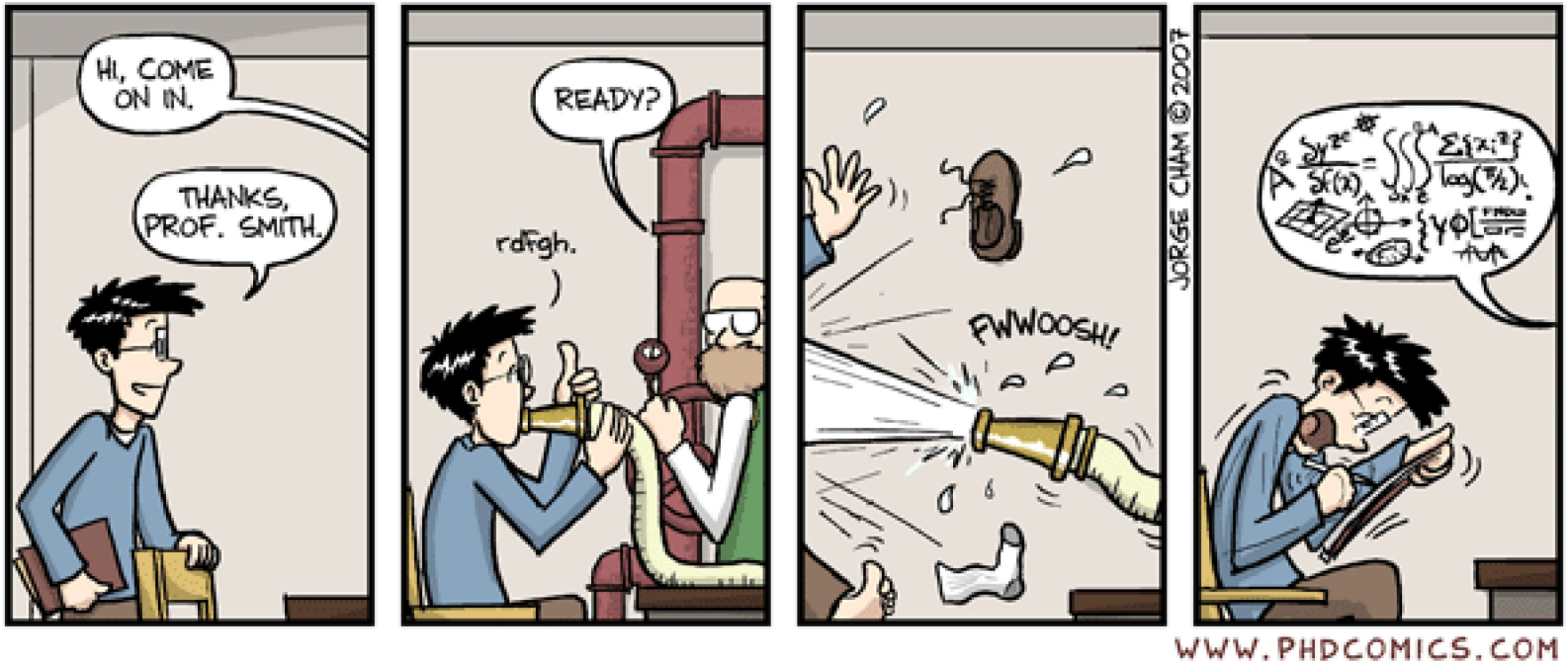
Peak volume of deals: 325k / s

Peak volume of payment: 256k / s

Peak volume in data base operation: 42000k / s

Streaming Data

Need to process data immediately while having strict memory constraints

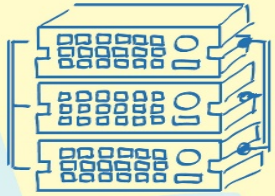


Chair of Process and Data Science

BIG DATA

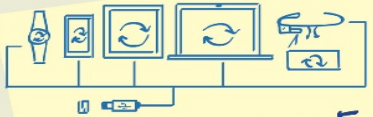
VOLUME

SCALE OF DATA



VARIETY

FORMS OF DATA



ANALYSIS OF DATA-FLOW

VELOCITY

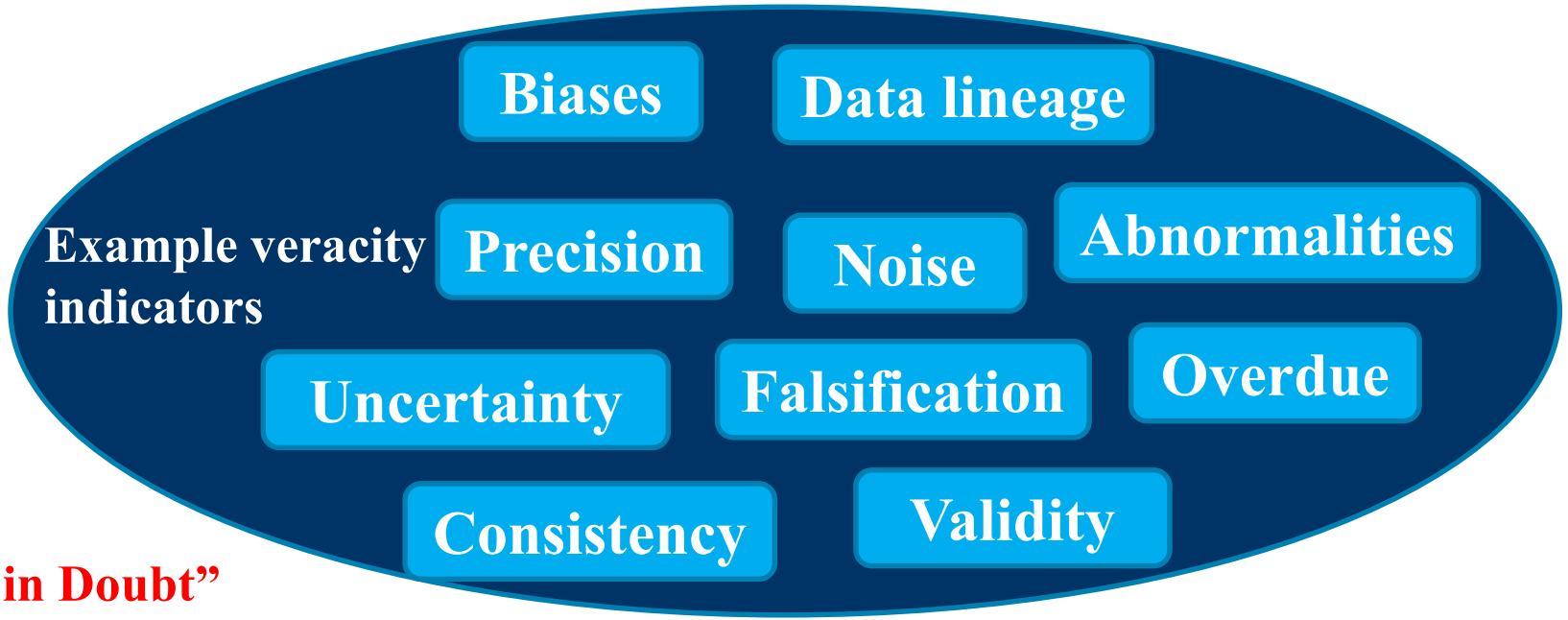
UNCERTAINTY OF DATA

VERACITY

Characteristics of big data

V4

- **Veracity = Uncertainty of Data**



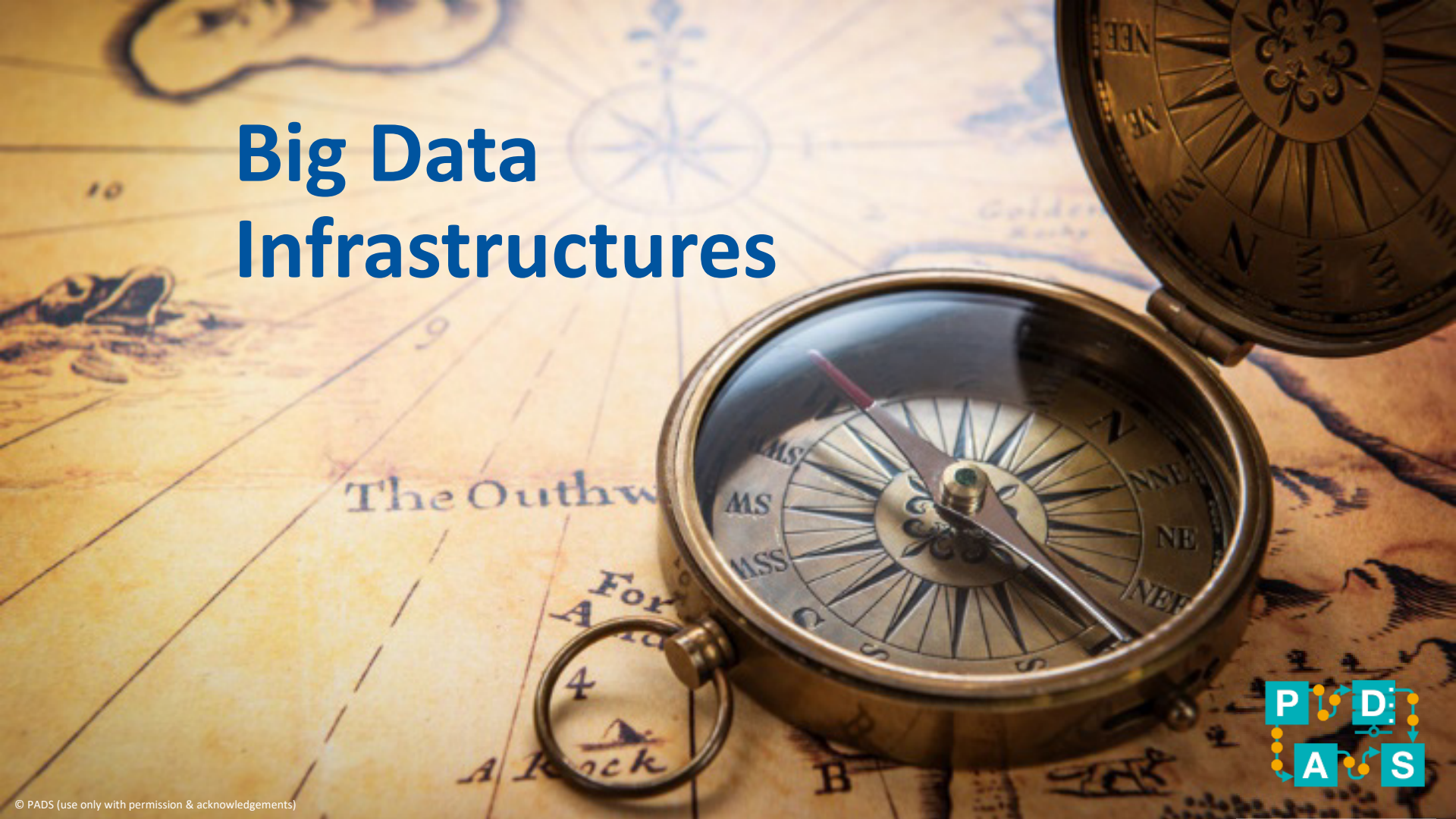
"Data in Doubt"

Are events recorded correctly?

V4

Accidental Alexa and Siri activations

Big Data Infrastructures



Big data infrastructures

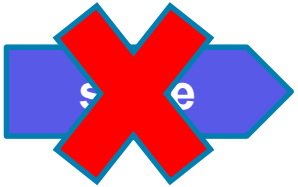
Motivations for the development of big data infrastructures

http://www.quuxlabs.com/wp-content/uploads/2010/08/Yahoo-hadoop-cluster_OSCON_2007.jpg

Big data infrastructures

We need to store and manage massive volumes of data (TB/EB/ZB)

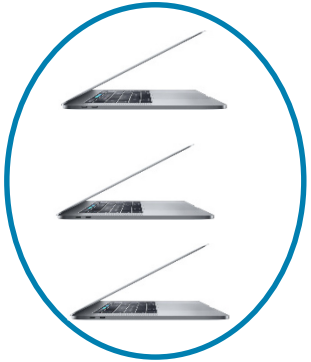
Hundreds of petabytes of data



4 TB SSD storage

Hundreds of petabytes of data

store



≠

Easy access and management of data

Big data infrastructures

We need to parallelize and distribute computation across hundreds or thousands of CPUs

Hundreds of petabytes of data

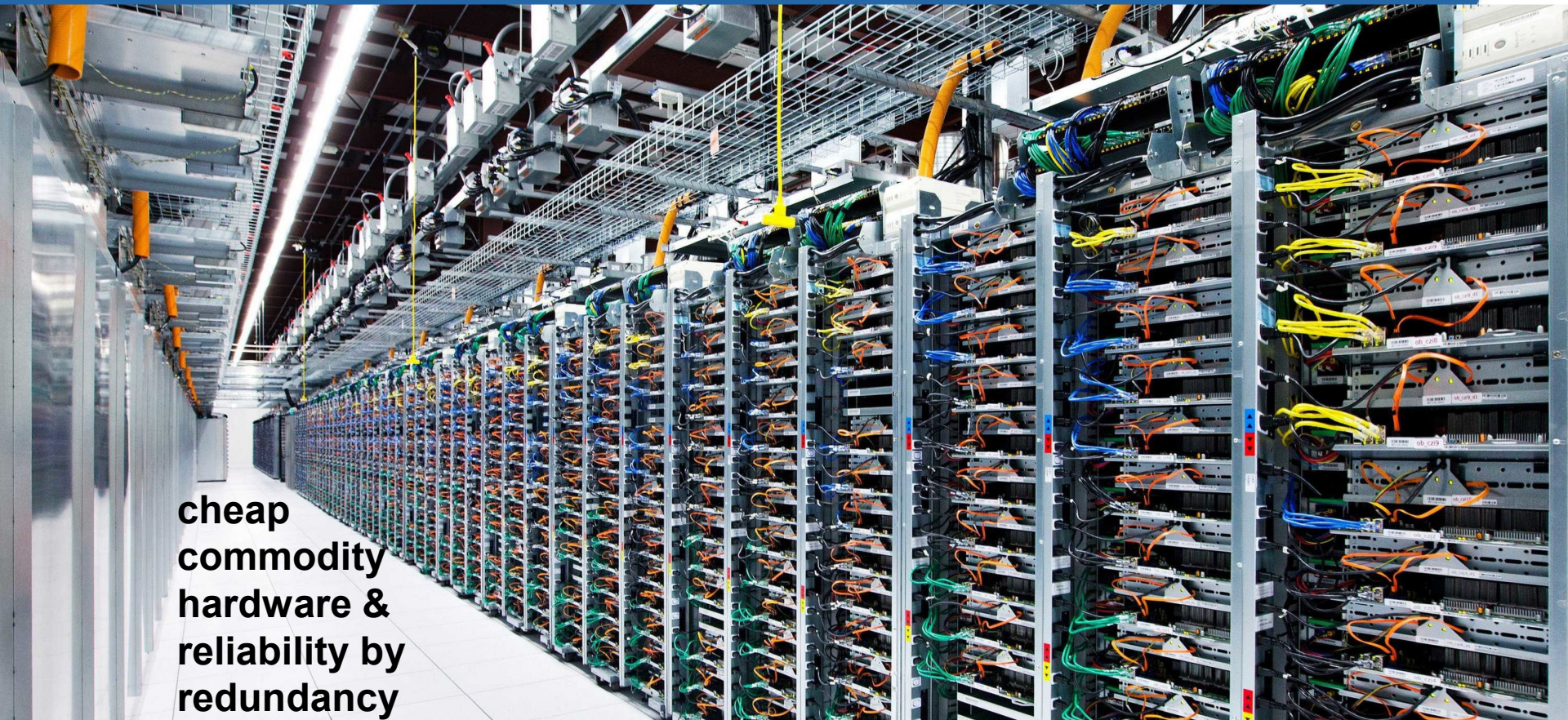
Hundreds of petabytes of data

process

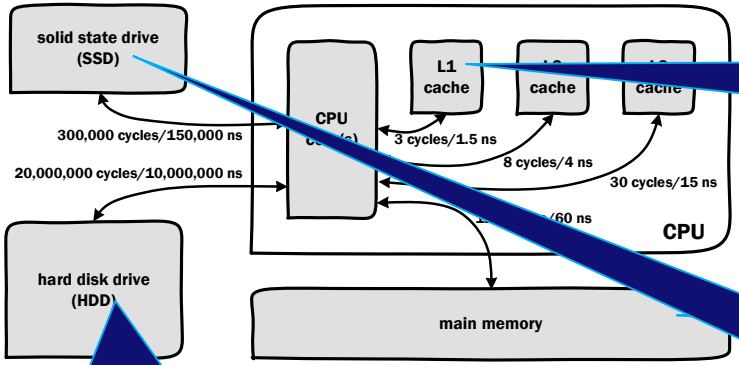
1000 x 

Chair of Process and Data Science

Google Data Centers



“Distances” in computing



getting a Nespresso from the kitchen

getting a coffee from the Starbucks around the corner

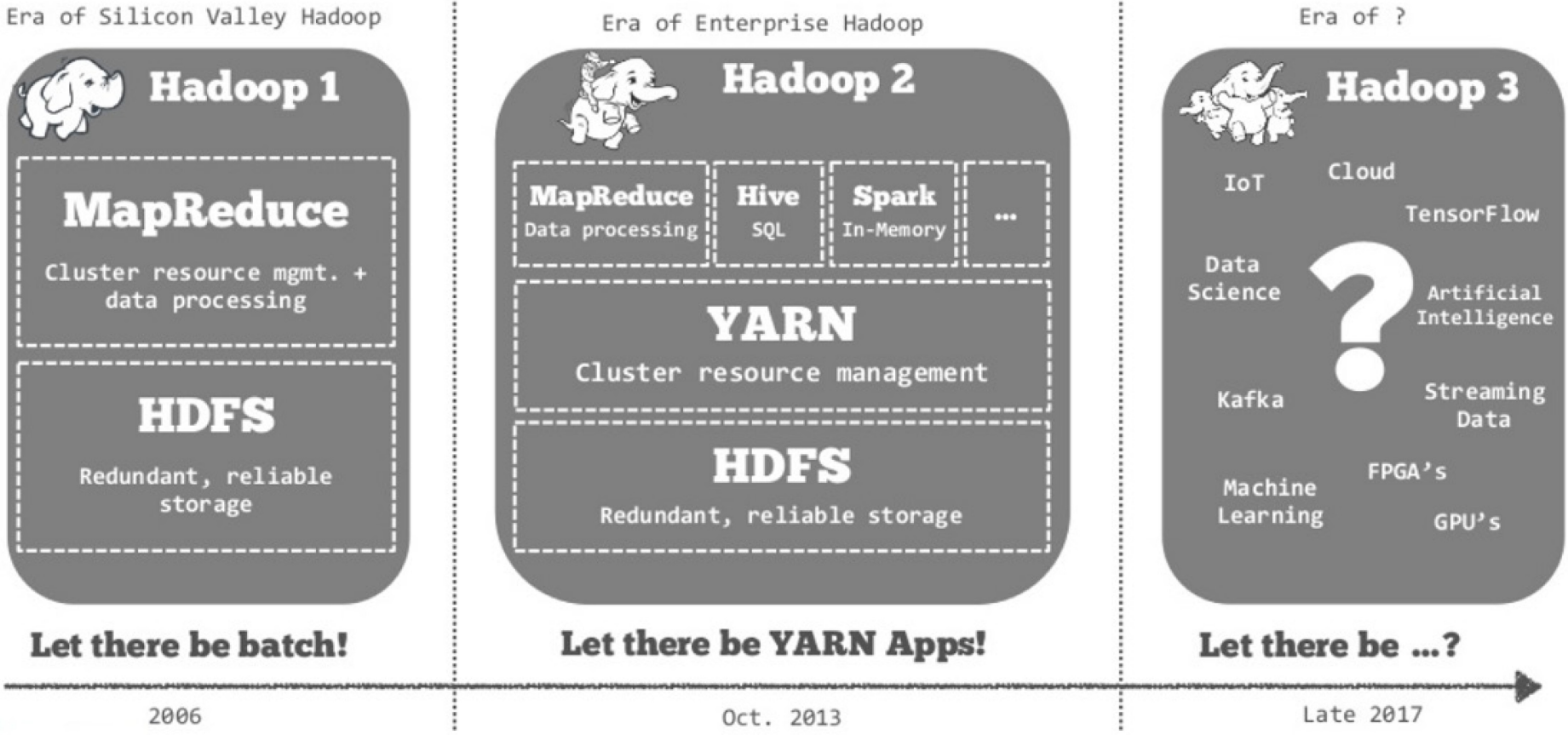
flying to Milano for a coffee

flying to Colombia, Ethiopia and Kenya, process the beans in Amsterdam, and then fly to Rome

Three Important Trends

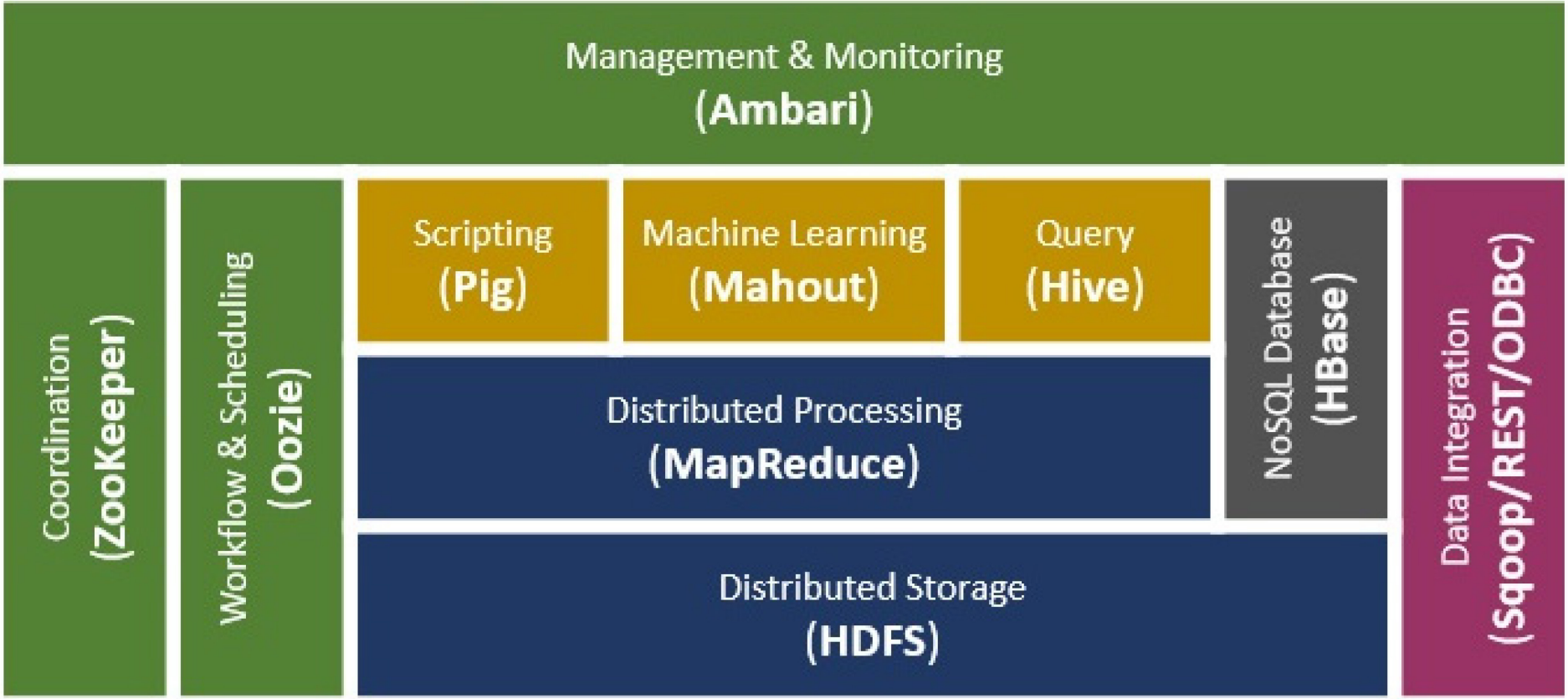
- More data in memory
- Distribution
  - Commodity hardware made reliable
  - Programming models like MapReduce
- Streaming

Hadoop History

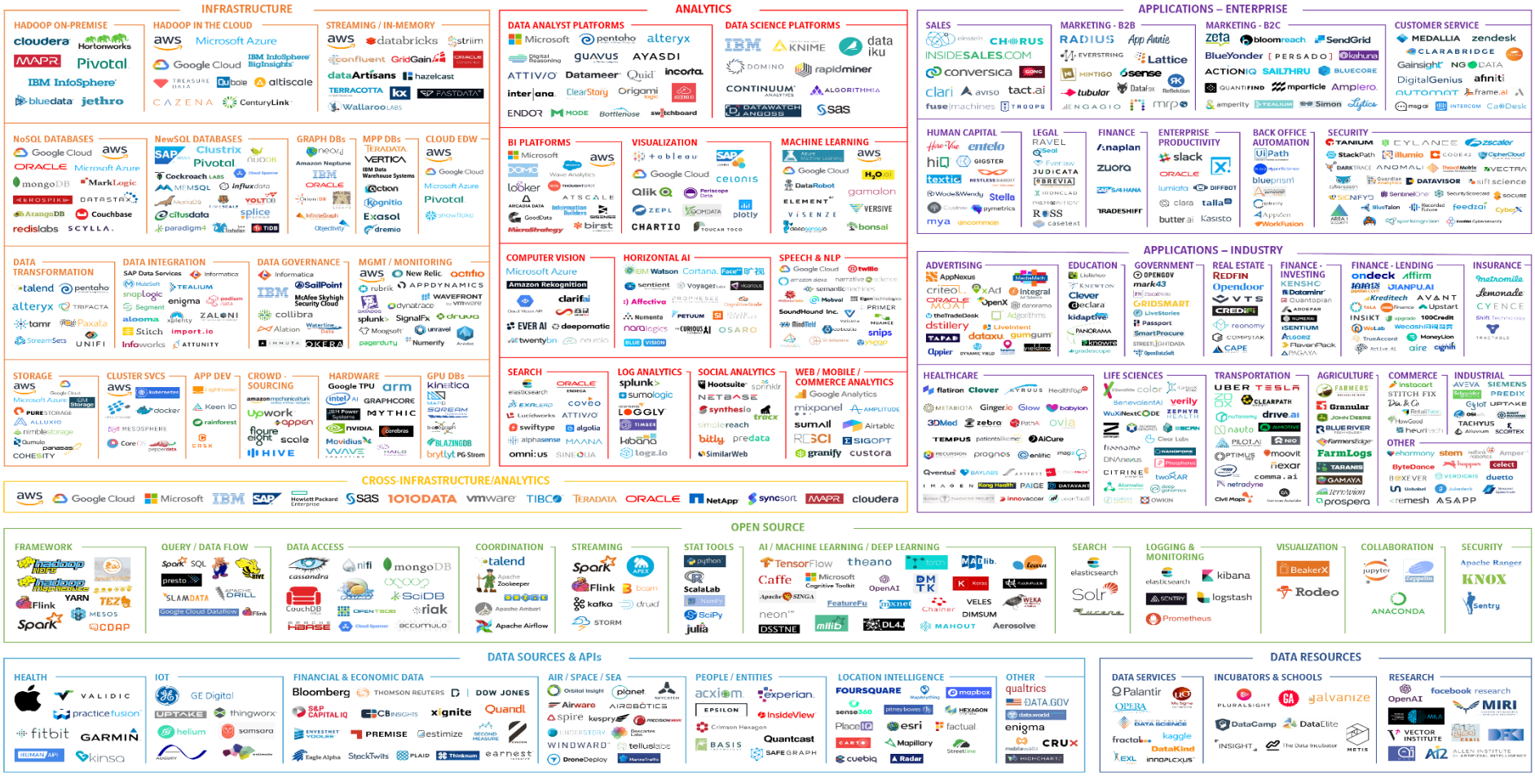


Apache Hadoop Ecosystem

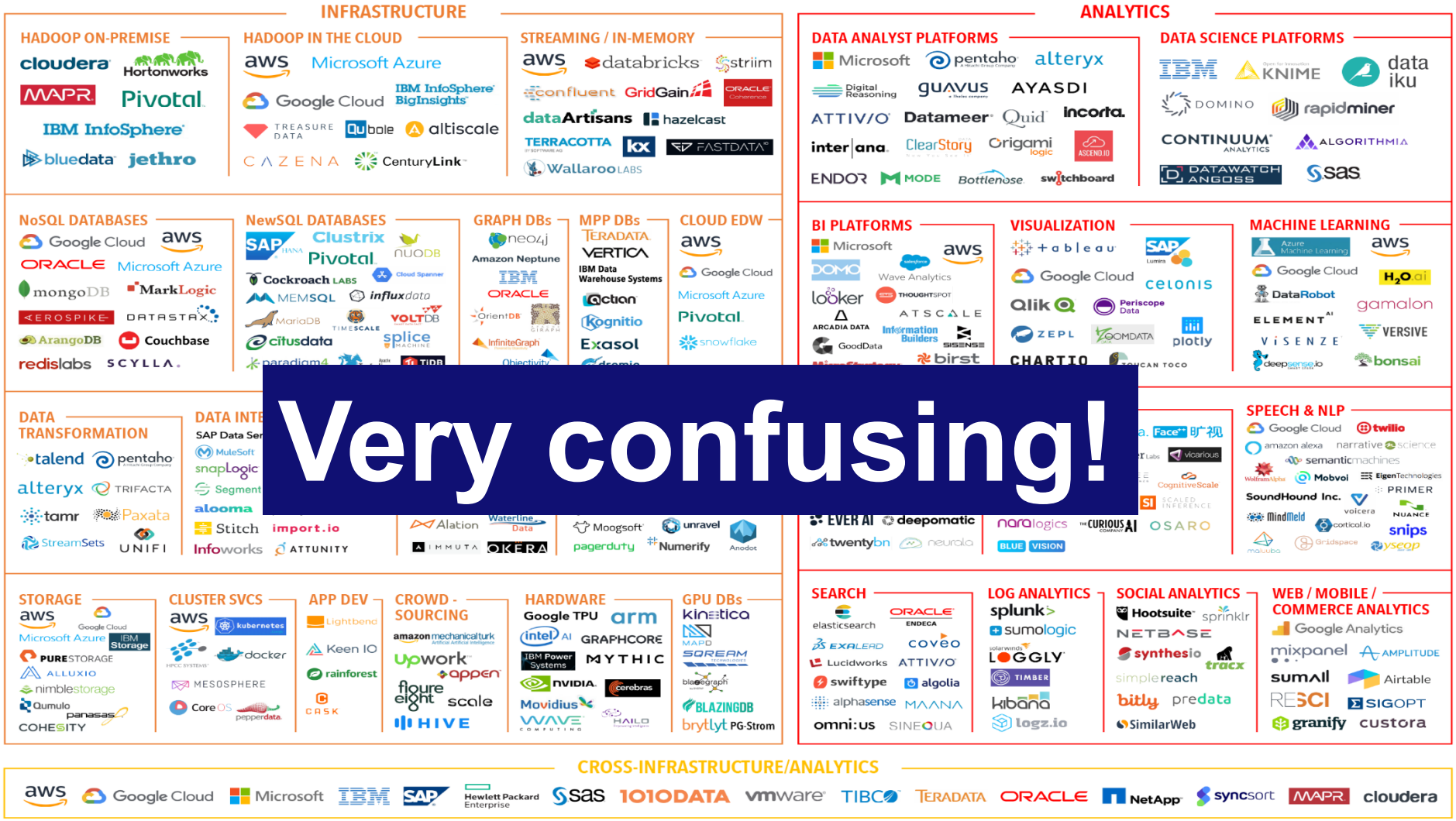
(main goal is to hide the complexity of distribution from the user and be fast and reliable)



BIG DATA & AI LANDSCAPE 2018



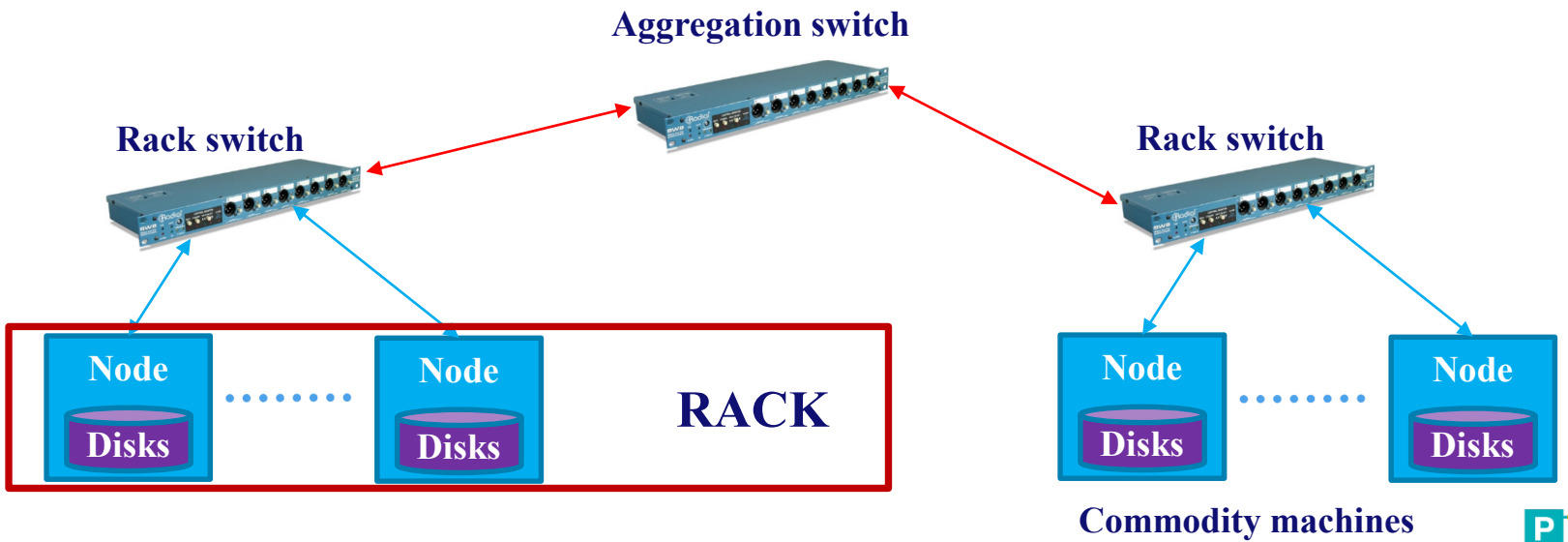
FIRSTMARK
EARLY STAGE VENTURE CAPITAL



Very confusing!

Distributed File System (DFS)

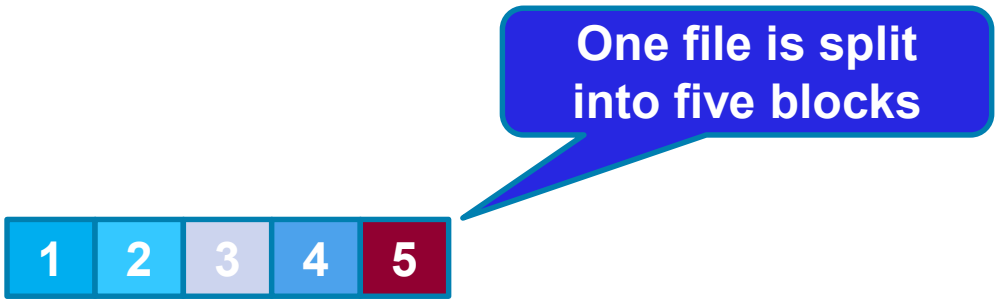
Distributed File System (DFS): The core of any Big data infrastructure



Chair of Process and Data Science

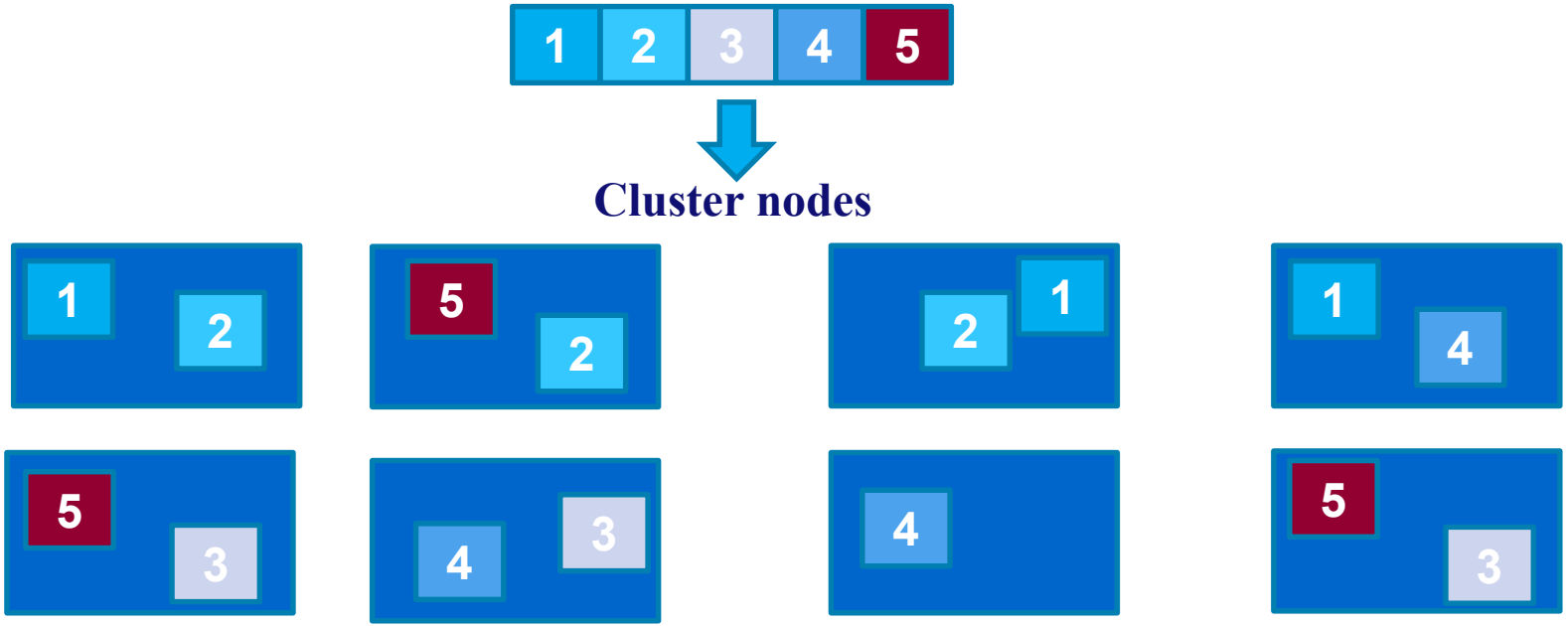
Distributed File System (DFS)

- **Distributed file system for big data**
  - **Each file written into DFS is split into data blocks**



Distributed File System (DFS)

- **Distributed file system for big data**
  - Each block is stored on one or more nodes (3 copies / block)



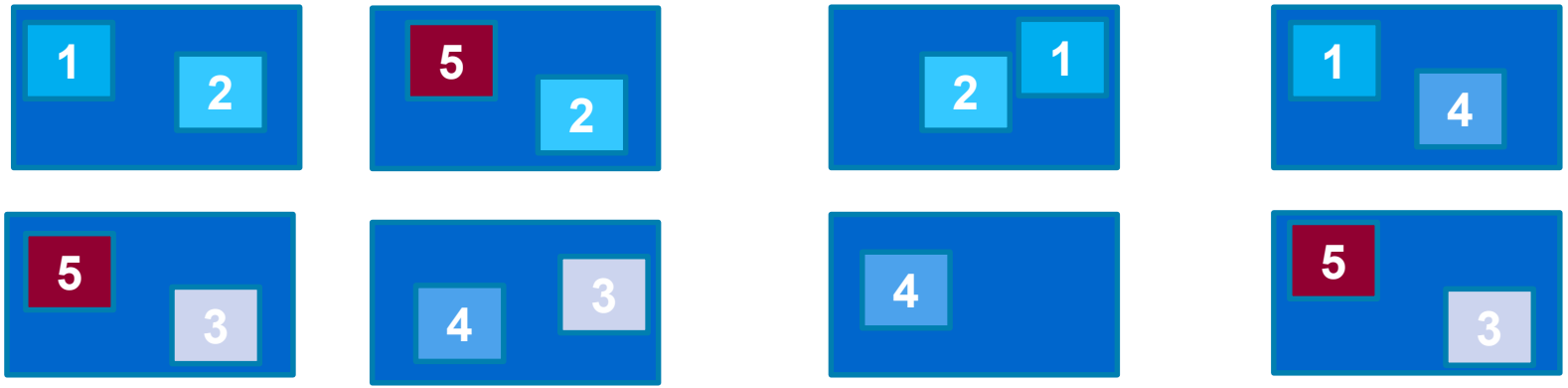
Distributed File System (DFS)

data distribution and replication

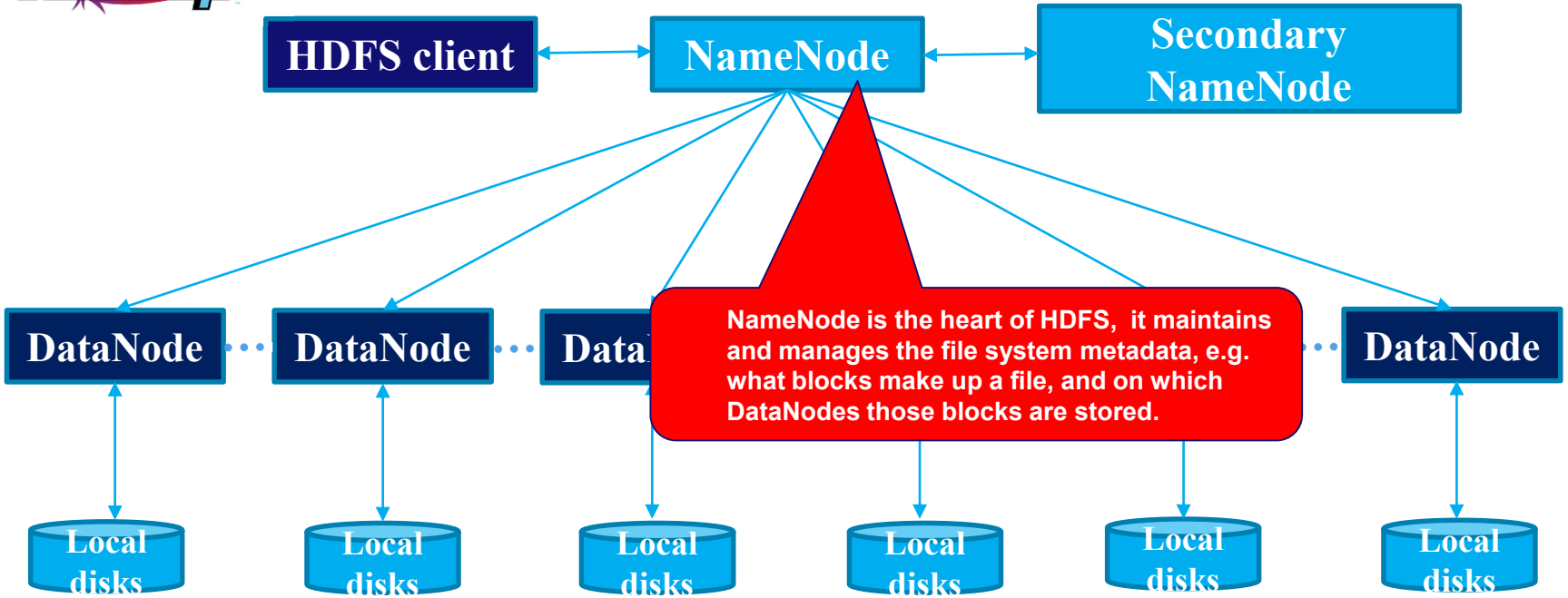
Assures

high concurrency
scalability
fault tolerance

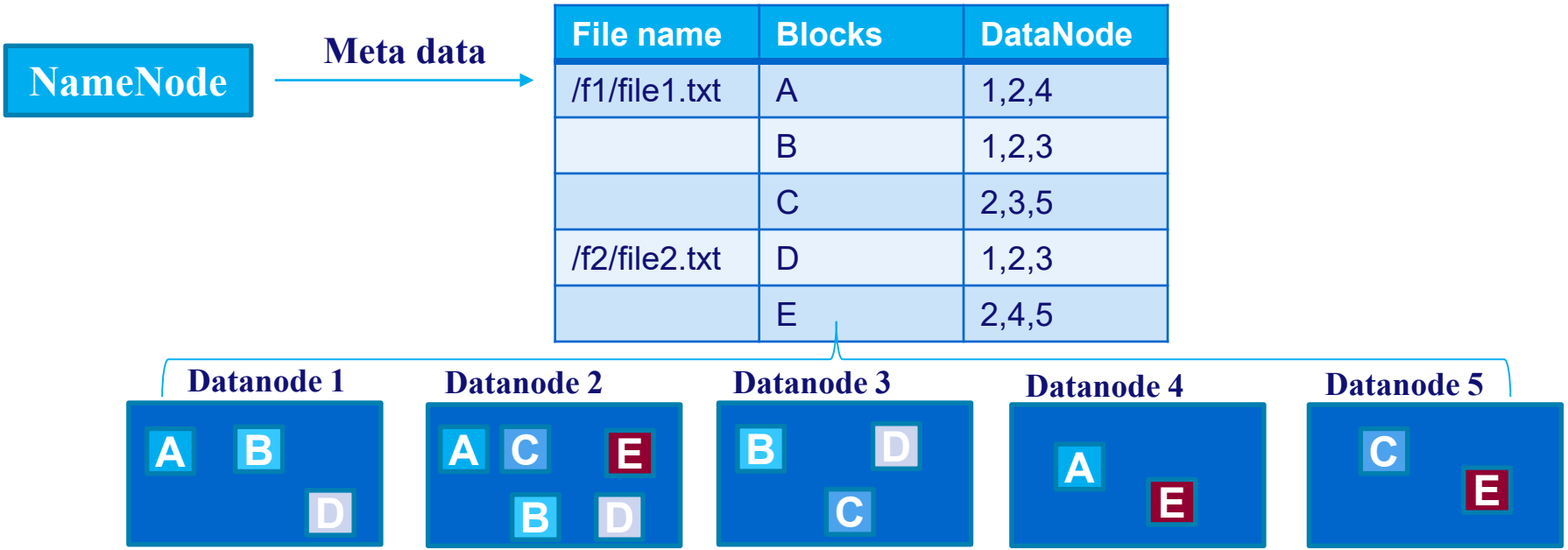
Cluster nodes



Hadoop Distributed File System (HDFS)



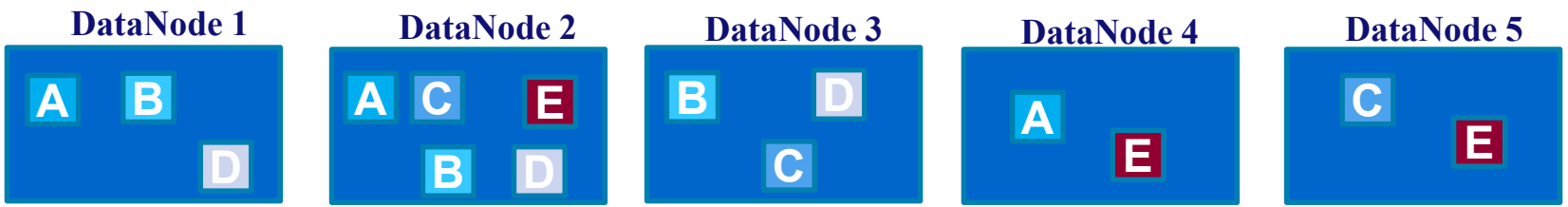
Hadoop Distributed File System (HDFS)



Chair of Process and Data Science

Hadoop Distributed File System (HDFS)

- **DataNode is where HDFS stores the actual data, there are usually many of such nodes.**



Chair of Process and Data Science

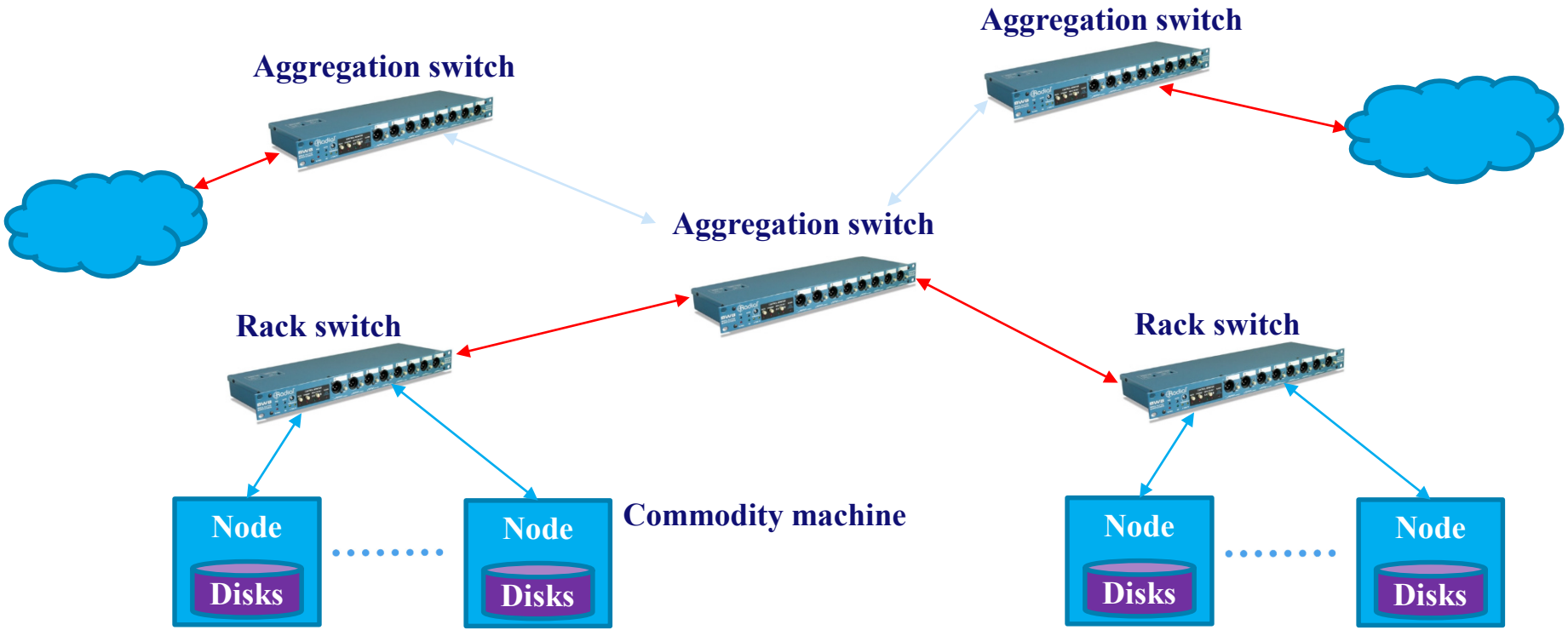
Hadoop Distributed File System (HDFS)

- HDFS is fault tolerant because the data are duplicated across multiple nodes to protect against machine failures.
- HDFS is highly scalable:
  - Data transfers happen directly with DataNodes so that read/write capacity scales well.
  - More capacity needed: just add more DataNodes and re-balance.

Distributed Computing Using a DFS

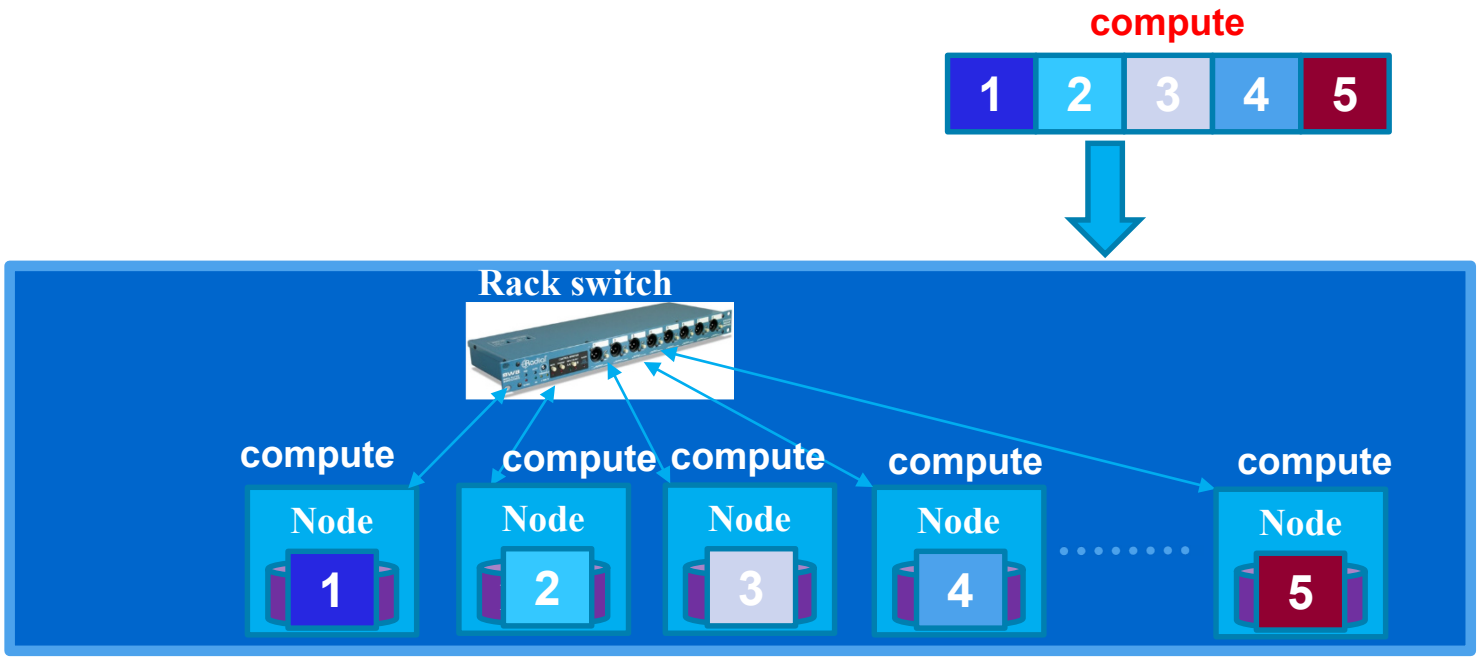
- Distributed system is composed of components that are located on different networked computers.
- These communicate and coordinate their actions by passing messages to achieve a common goal.

Distributed Computing Using a DFS



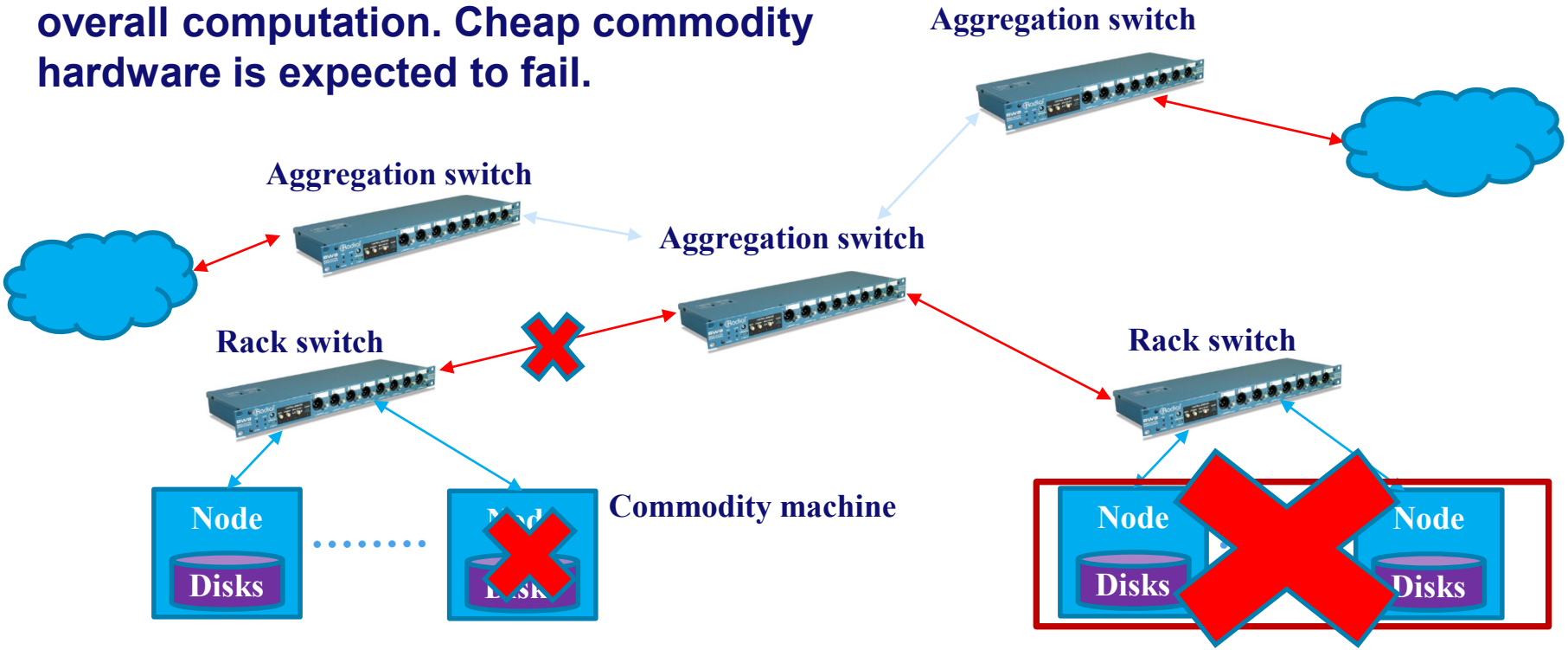
Distributed Computing Using a DFS

Data parallelism: Move computation to data!



Distributed Computing Using a DFS

Fault-tolerant: No need to restart the overall computation. Cheap commodity hardware is expected to fail.

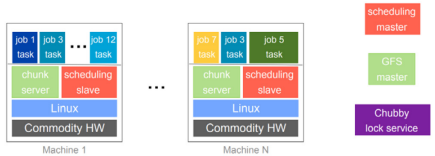


Some numbers from Jeff Dean (Google)

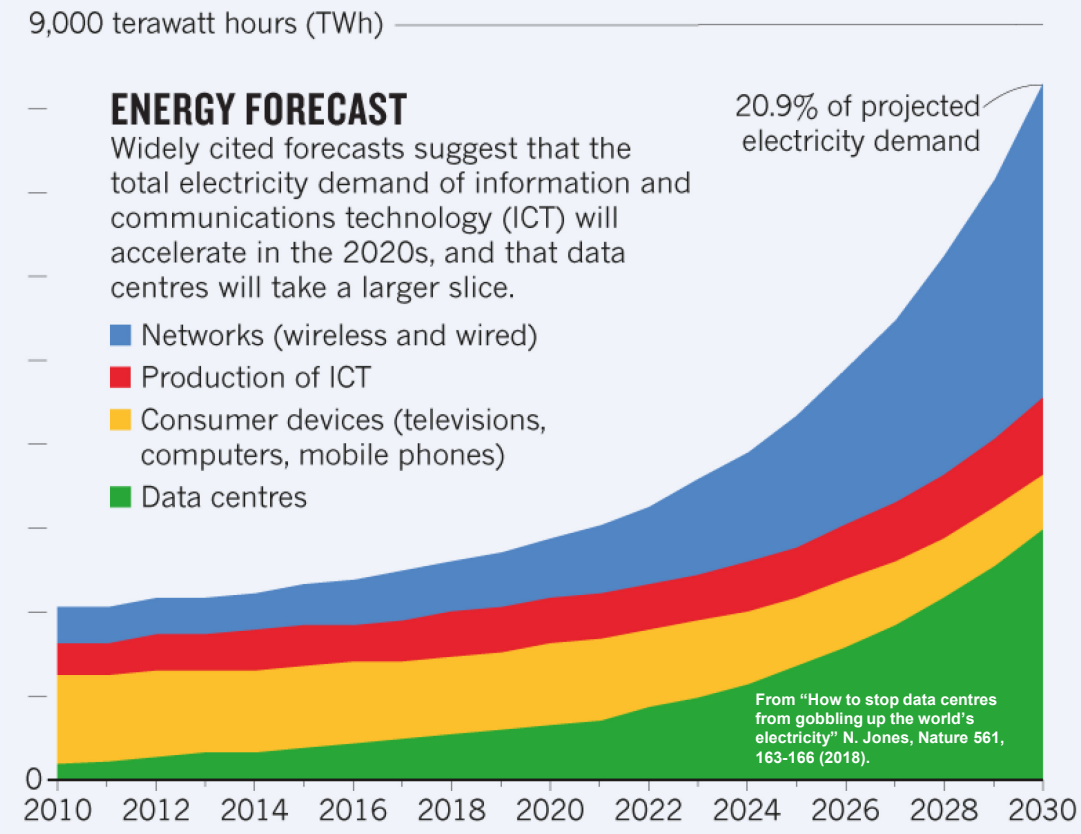
Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
~1 network rewiring (rolling ~5% of machines down over 2-day span)
~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
~5 racks go wonky (40-80 machines see 50% packetloss)
~8 network maintenances (4 might cause ~30-minute random connectivity losses)
~12 router reloads (takes out DNS and external vips for a couple minutes)
~3 router failures (have to immediately pull traffic for an hour)
~dozens of minor 30-second blips for dns
~1000 individual machine failures
~thousands of hard drive failures
slow disks, bad memory, misconfigured machines, flaky machines, etc.

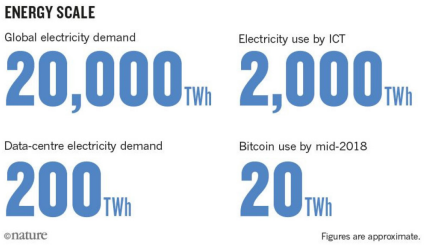
- Cluster is 1000s of machines, typically one or handful of configurations
- File system (GFS) + Cluster scheduling system are core services
- Typically 100s to 1000s of active jobs (some w/1 task, some w/1000s)
  - mix of batch and low-latency, user-facing production jobs



Energy consumption data centers

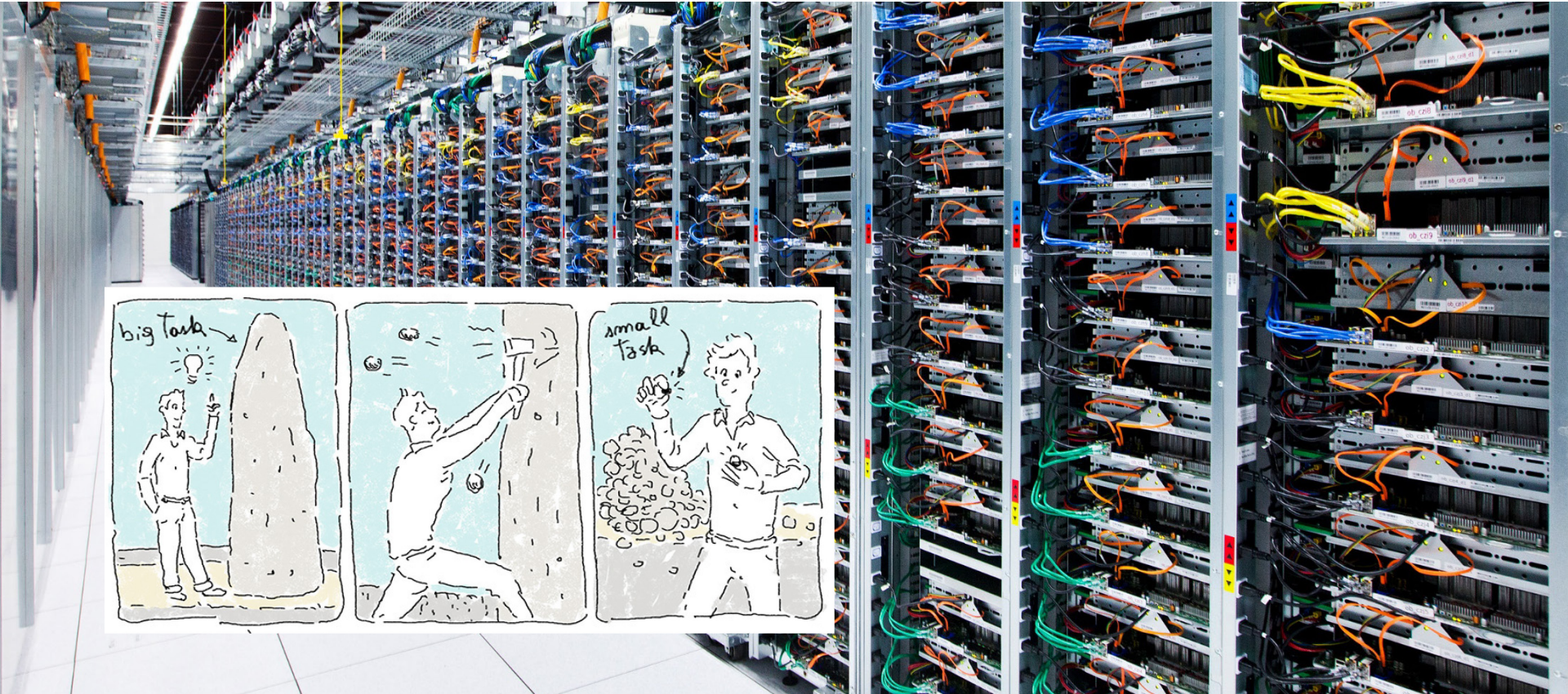


In this Google-owned data center, blue pipes supply cold water and red pipes return warm water to be cooled.



P D A S Chair of Process and Data Science

How to exploit concurrency?



MapReduce

MapReduce (2004)

- MapReduce is a programming model.
- Speed-up through distribution.
- Part of Apache Hadoop.
- Google has the MapReduce patent (but not really new).

MAPREDUCE: SIMPLIFIED DATA PROCESSING ON LARGE CLUSTERS

by Jeffrey Dean and Sanjay Ghemawat

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of a *map* and a *reduce* function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks. Programmers find the system easy to use: more than ten thousand distinct MapReduce programs have been implemented internally at Google over the past four years, and an average of one hundred thousand MapReduce jobs are executed on Google's clusters every day, processing a total of more than twenty petabytes of data per day.

1 Introduction

Prior to our development of MapReduce, the authors and many others at Google implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, Web request logs, etc., to compute various kinds of derived data, such as inverted indices, various representations of the graph structure of Web documents, summaries of the number of pages crawled per host, and the set of most frequent queries in a given day. 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 failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a map operation to each logical record in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key in order to combine the derived data appropriately. Our use of a functional model with user-specified map and reduce operations allows us to parallelize large computations easily and to use re-execution as the primary mechanism for fault tolerance.

The major contributions of this work are a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs. The programming model can also be used to parallelize computations across multiple cores of the same machine.

Section 2 describes the basic programming model and gives several examples. In Section 3, we describe an implementation of the MapReduce interface tailored towards our cluster-based computing environment. Section 4 describes several refinements of the programming model that we have found useful. Section 5 has performance measurements of our implementation for a variety of tasks. In Section 6, we explore the use of MapReduce within Google including our experiences in using it as the basis for a rewrite of our production indexing system. Section 7 discusses related and future work.

2 Programming Model

The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the MapReduce library expresses the computation as two functions: map and reduce.

Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the reduce function.

The reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges these values together to form a possibly smaller set of values. Typically just zero or one output value is produced per reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory.

2.1 Example

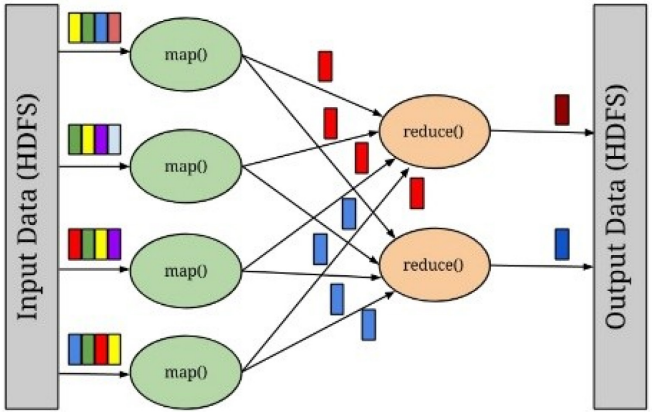
Consider the problem of counting the number of occurrences of each word in a large collection of documents. The user would write code similar to the following pseudocode.

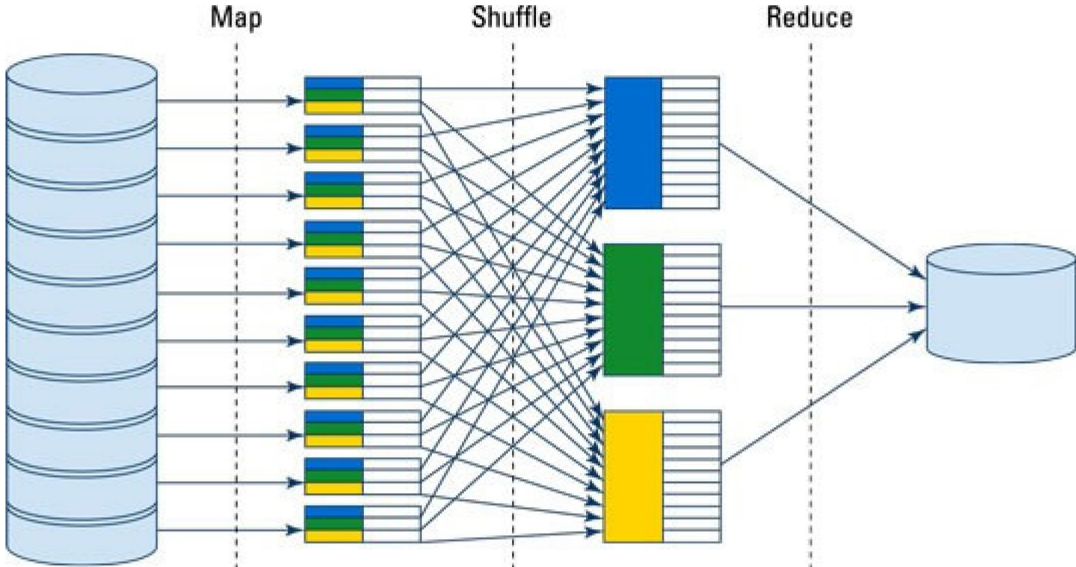
Biographies

Jeff Dean (jeff@google.com) is a Google fellow and is currently working on a large variety of large-scale distributed systems at Google's Mountain View, CA, facility.

Sanjay Ghemawat (sanjay@google.com) is a Google Fellow and works on the distributed computing infrastructure used by most the company's products. He is based at Google's Mountain View, CA, facility.

Main idea

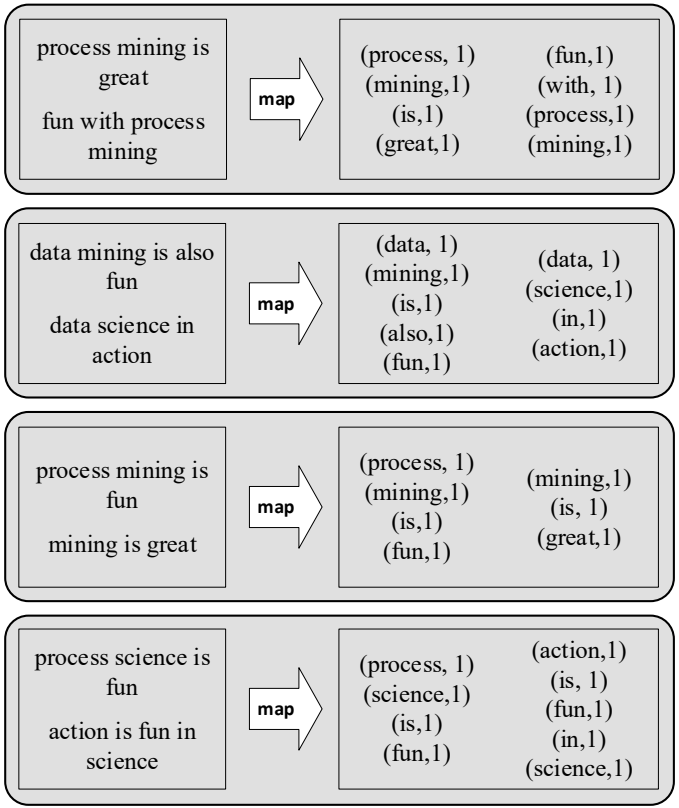

Input Data (HDFS)
map()
map()
map()
map()
reduce()
reduce()
Output Data (HDFS)


Map
Shuffle
Reduce

MapReduce

- K_1, K_2 are key domains.
- V_1, V_2, V_3 are value domains.
- $map \in K_1 \times V_1 \rightarrow (K_2 \times V_2)^*$ is the map function.
  - The map function is applied in parallel to every pair (keyed by K_1) in the input dataset.
  - This produces a list of pairs (keyed by K_2) for each call.
- $reduce \in K_2 \times (V_2)^* \rightarrow (V_3)^*$ is the reduce function.
  - The reduce function is applied to the grouped values (keyed by K_2).
  - Often the result is just a single value (keyed by K_2).
- The infrastructure takes care of the shuffle connecting sets of $(K_2 \times V_2)^*$ values to sets of $K_2 \times (V_2)^*$ values in a distributed environment.

Counting words: Map step

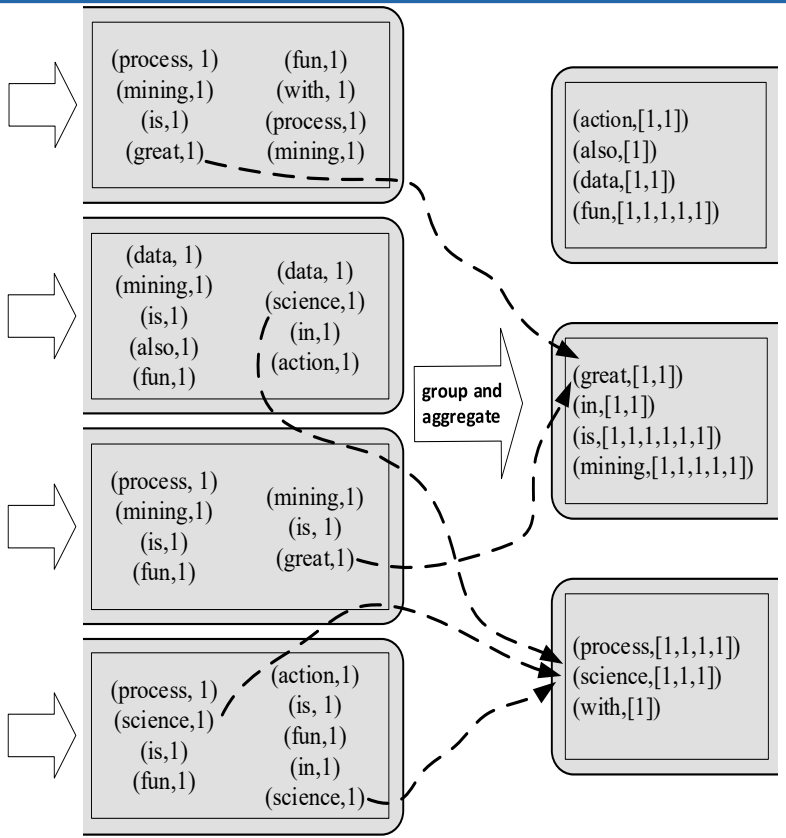


$$map \in K_1 \times V_1 \rightarrow (K_2 \times V_2)^*$$

Distributed!

Counting words: Shuffle step

Result of Map step

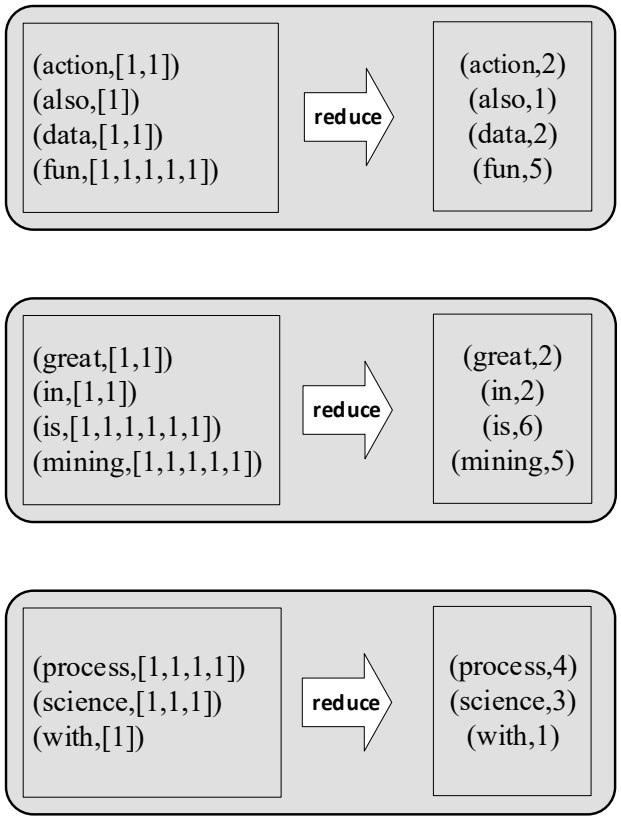


Input for Reduce step

Chair of Process and Data Science

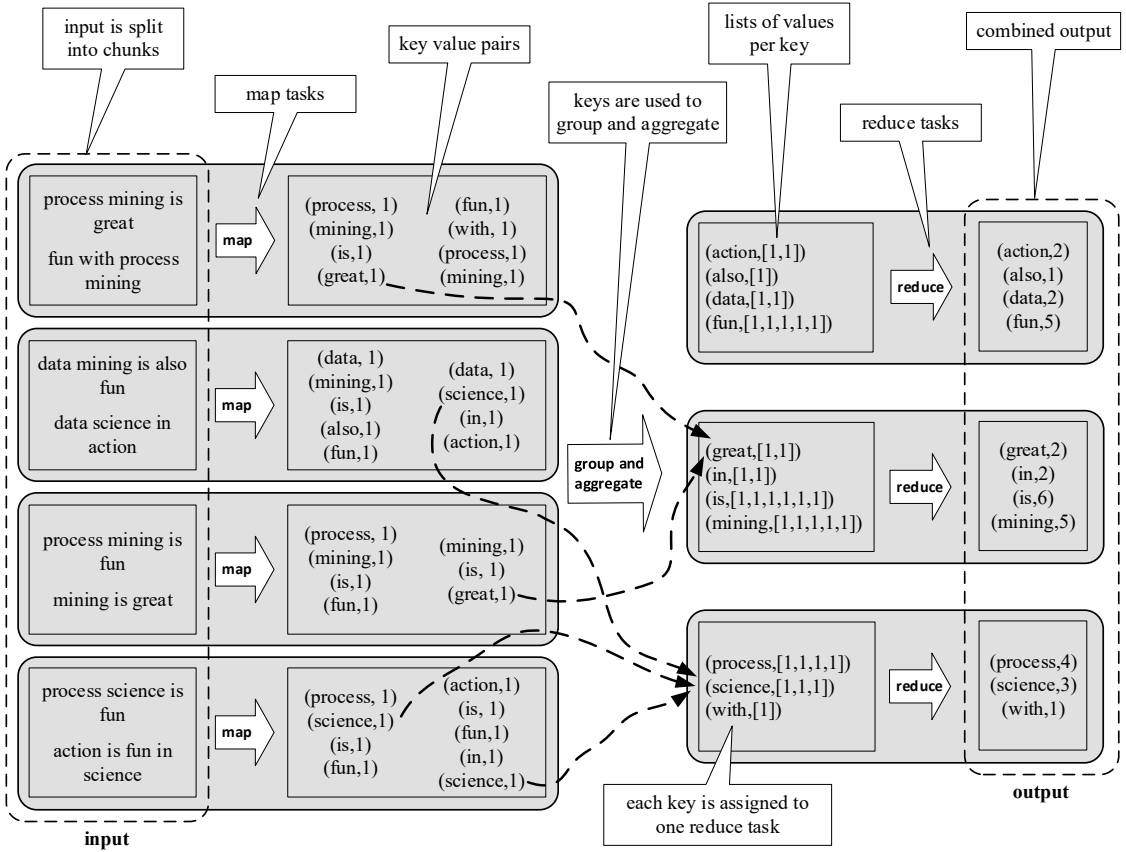
Counting words: Reduce step

$$reduce \in K_2 \times (V_2)^* \rightarrow (V_3)^*$$

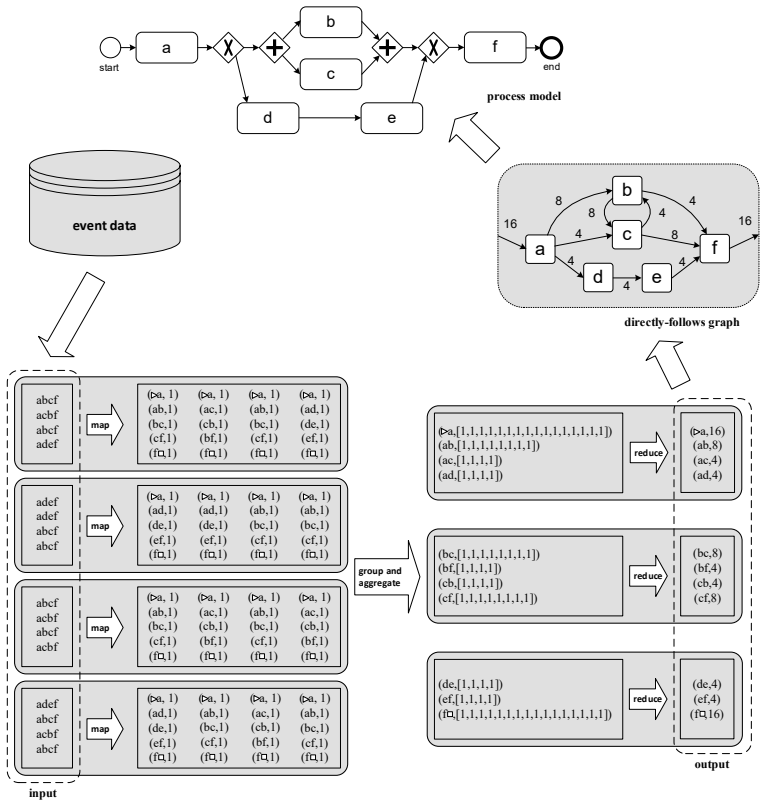


Distributed!

Overview



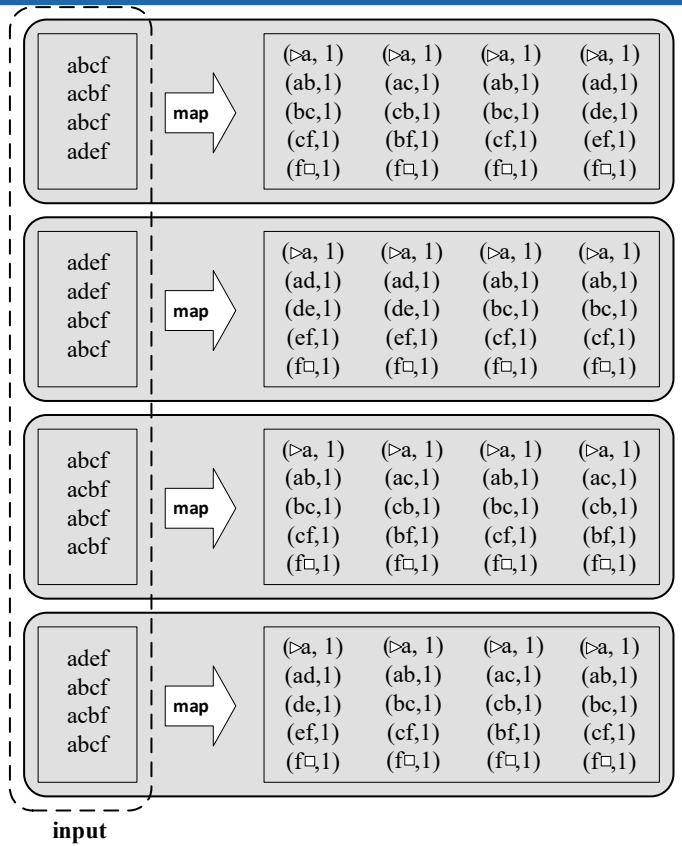
MapReduce applied to process discovery



- We need to count how many times activity x is followed by y in an event log with billions of events.
- Assume traces are split over multiple nodes.

Chair of Process and Data Science

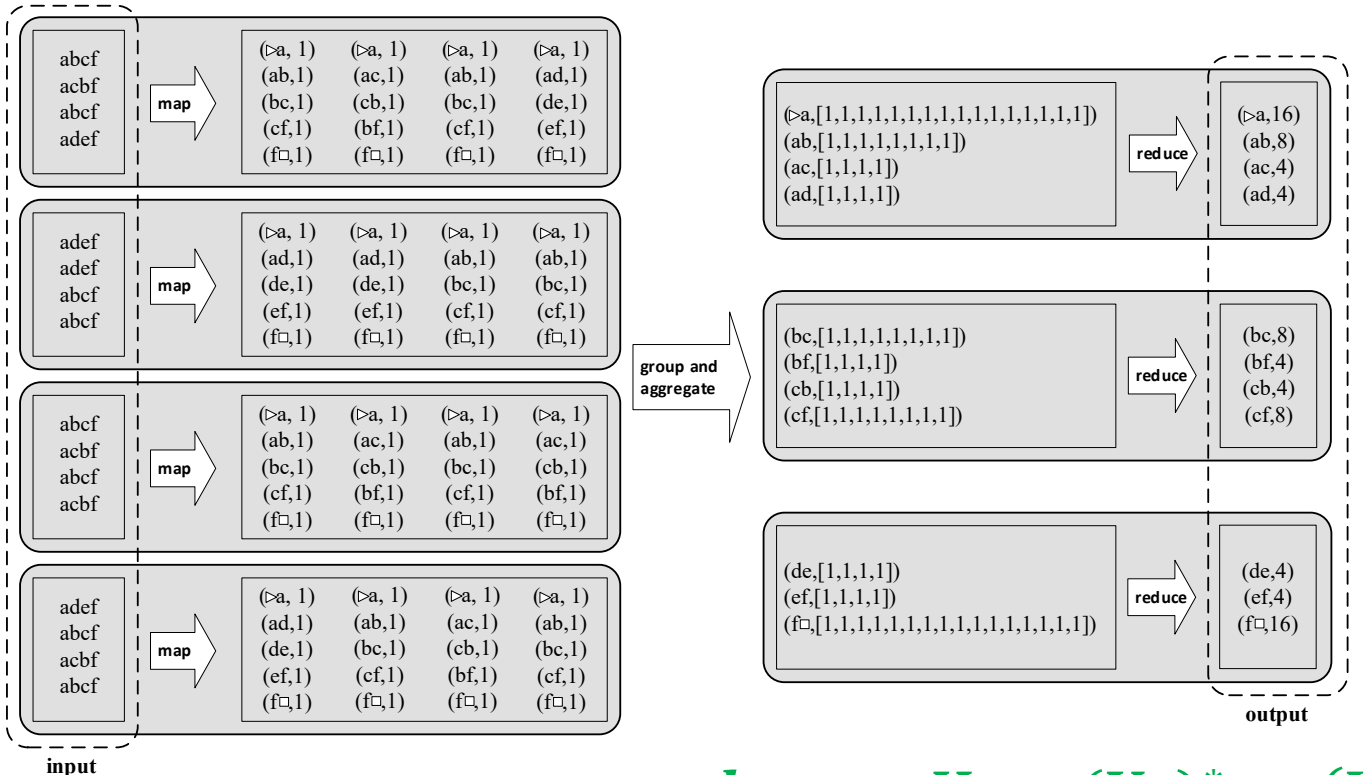
MapReduce applied to process discovery



$$map \in K_1 \times V_1 \rightarrow (K_2 \times V_2)^*$$

Chair of Process and Data Science

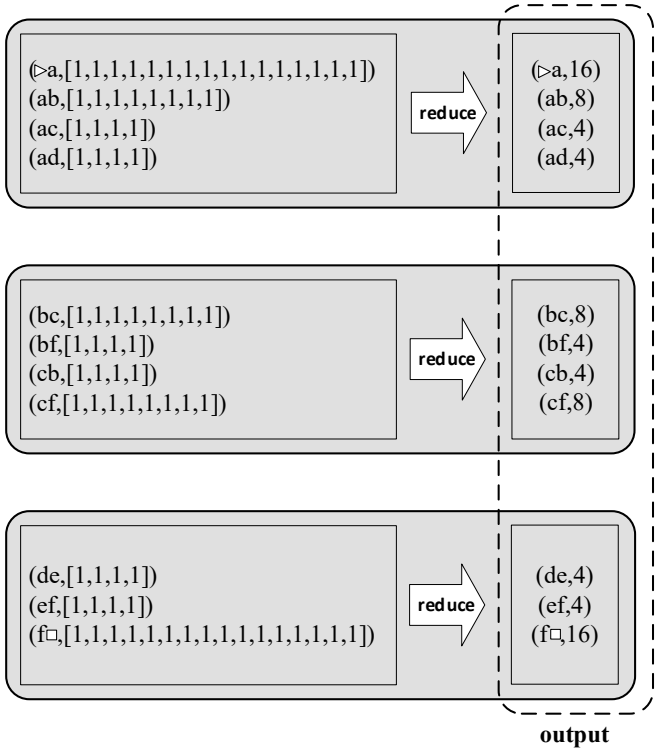
MapReduce applied to process discovery

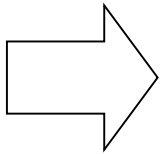


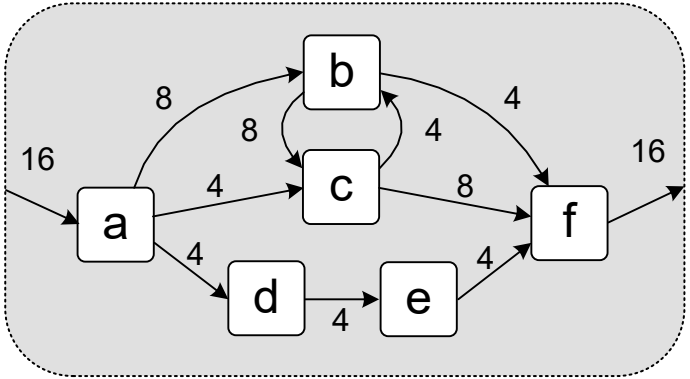
$$reduce \in K_2 \times (V_2)^* \rightarrow (V_3)^*$$

Chair of Process and Data Science

MapReduce applied to process discovery



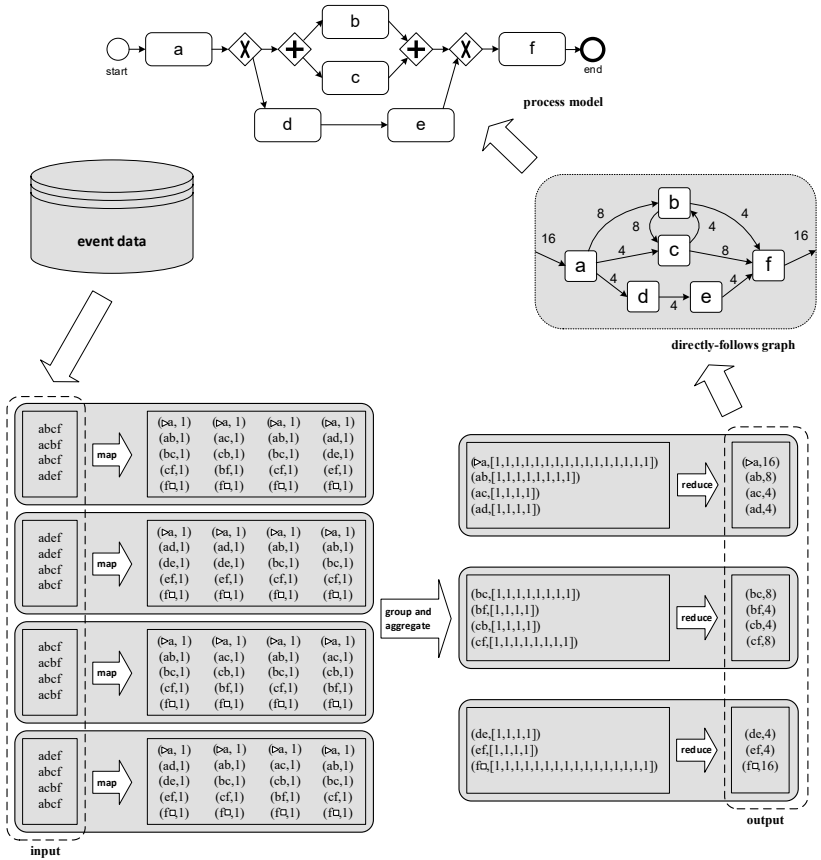




directly-follows graph

Chair of Process and Data Science

MapReduce applied to process discovery



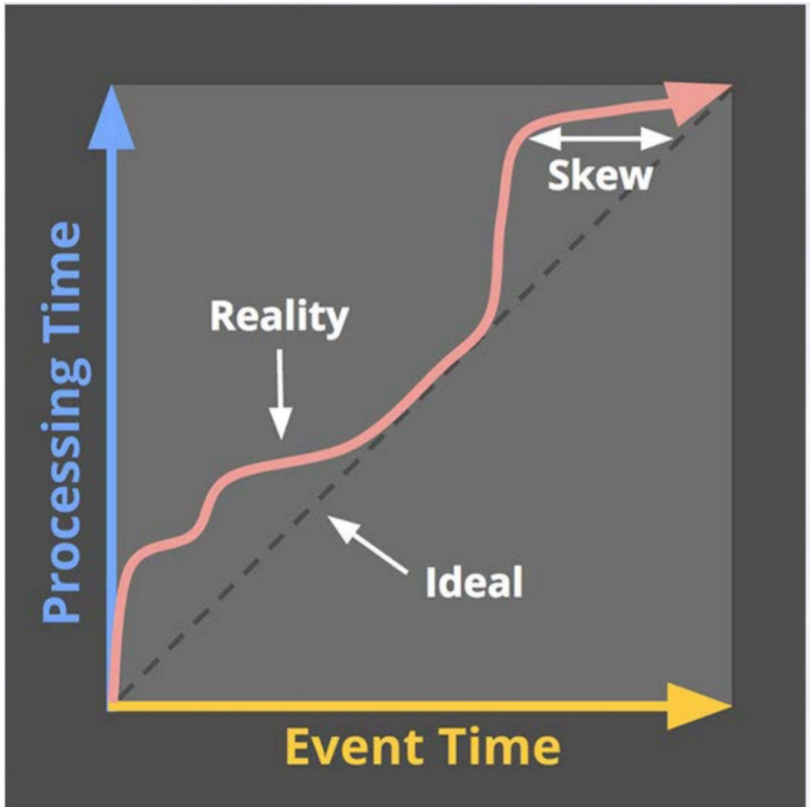
This way process discovery is scalable at the scale of Google (provided you have enough hardware).

Chair of Process and Data Science

Streaming

Streaming

Streaming: Respond “immediately”



- Event time: the time at which the data object occurred
- Processing time: the time at which the data object is observed in streaming system

https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101

Streaming: Limited capacity

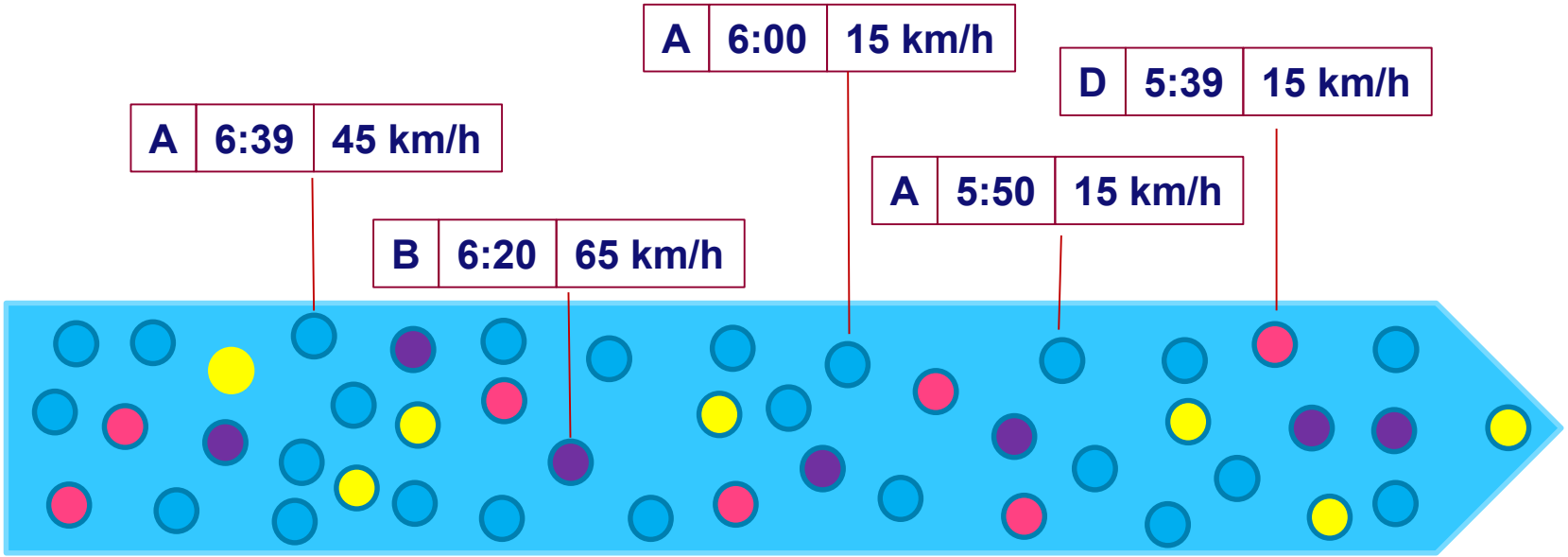
- Limited capacity
- When “new data” get stored, “old data” need to be removed.

Streaming Examples

- **Stock market surveillance** (e.g., predict the trend of stock).
- Identification of life-threatening **conditions in health care** (e.g., continuously monitor the physical signs of patients and alert when anomaly happens).
- **Computer system and network monitoring** (e.g., monitor and analyze the amount of packets sent from the IP addresses that access web server).
- Dealing with data stream from various kinds of **sensors** (e.g., continuously analyze the sensors in aircrafts for anomalies).

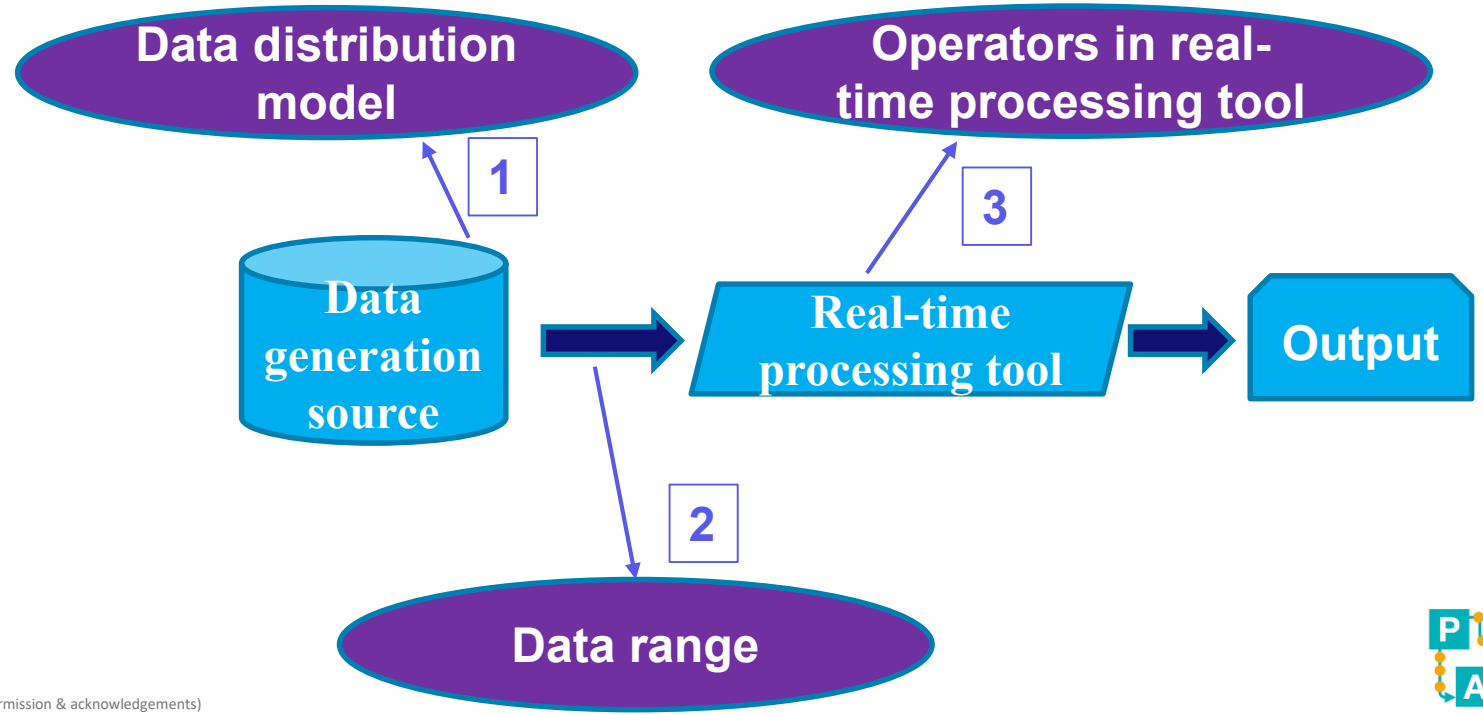
Streaming

- **An example data stream coming from cars that are monitored**



Streaming

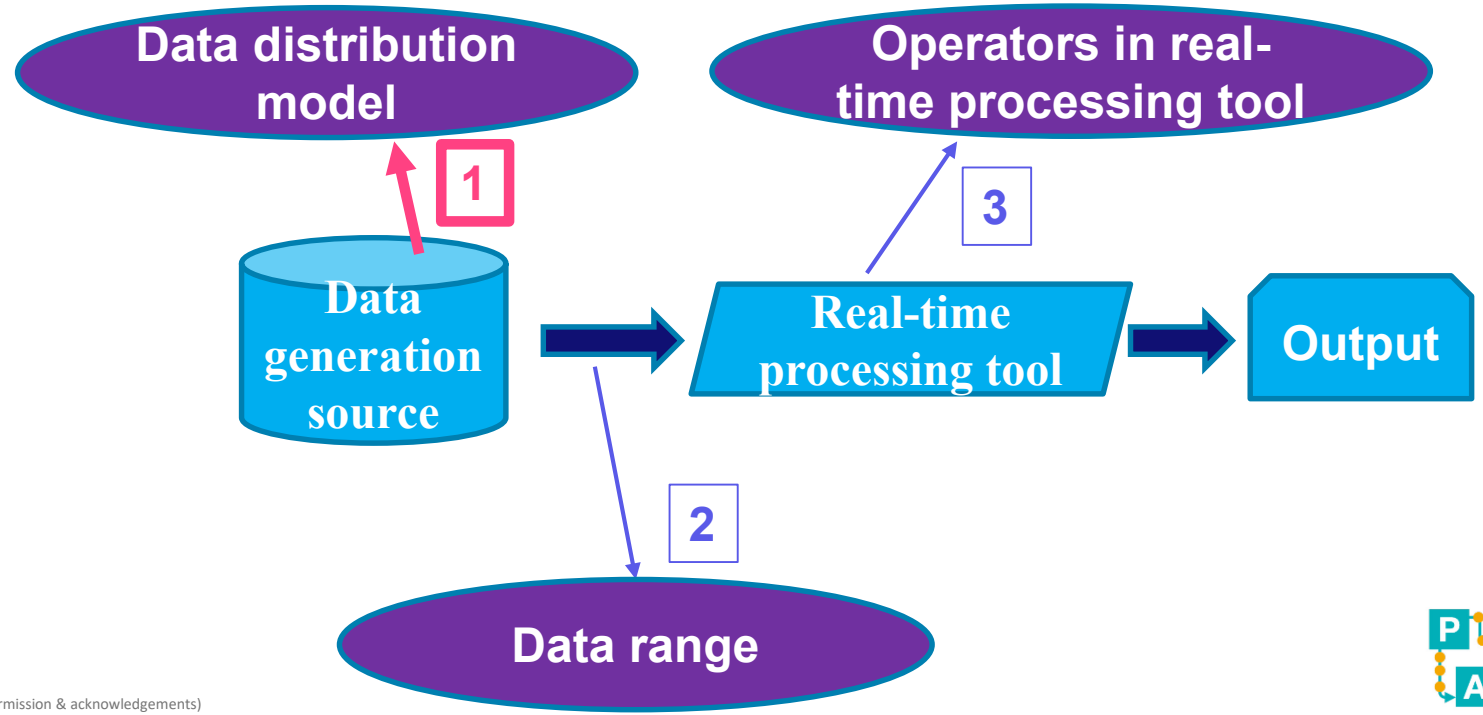
- **Important elements in streaming**



Chair of Process and Data Science

Streaming

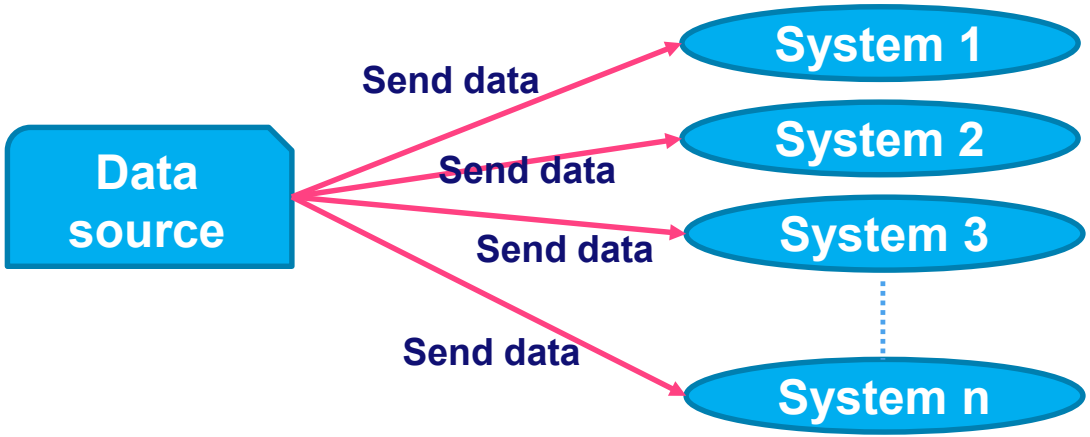
- **Important elements in streaming**



Chair of Process and Data Science

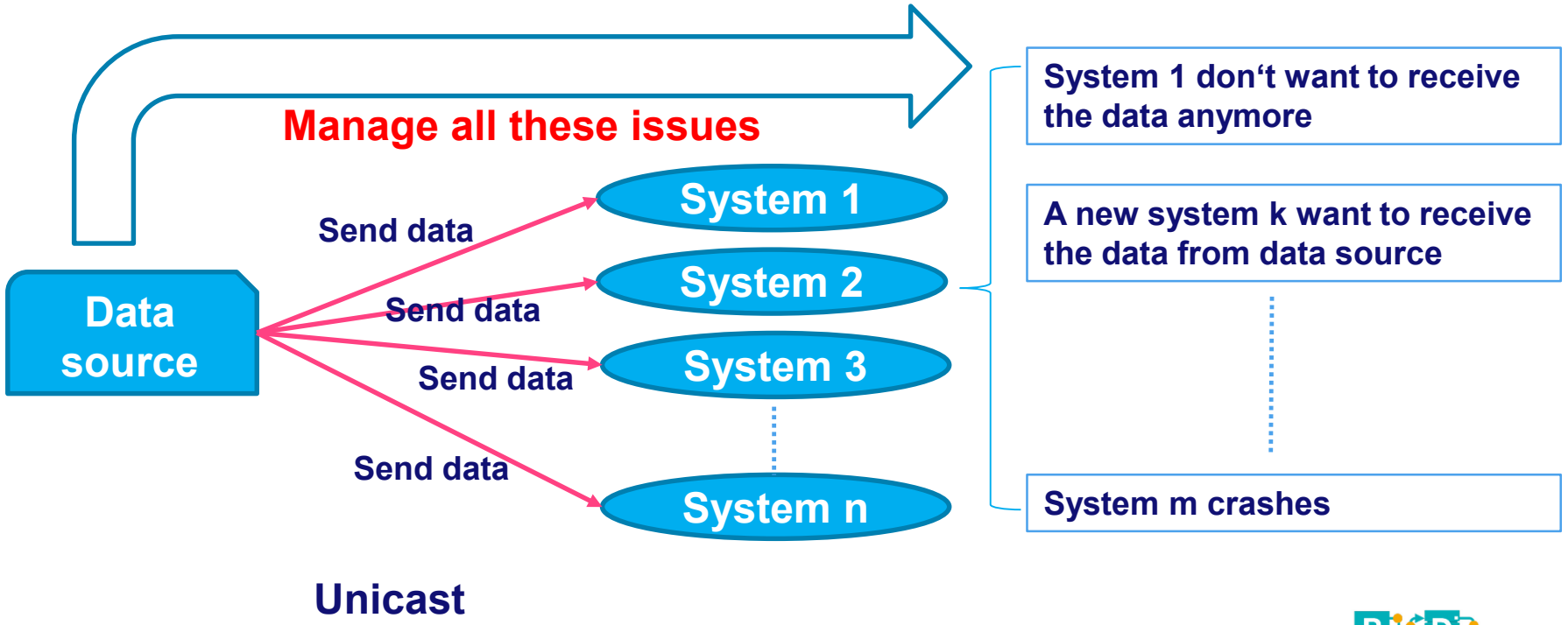
Streaming

- **Data distribution model**
  - **Why not unicast?**



Unicast (one-to-one transmission)

Streaming

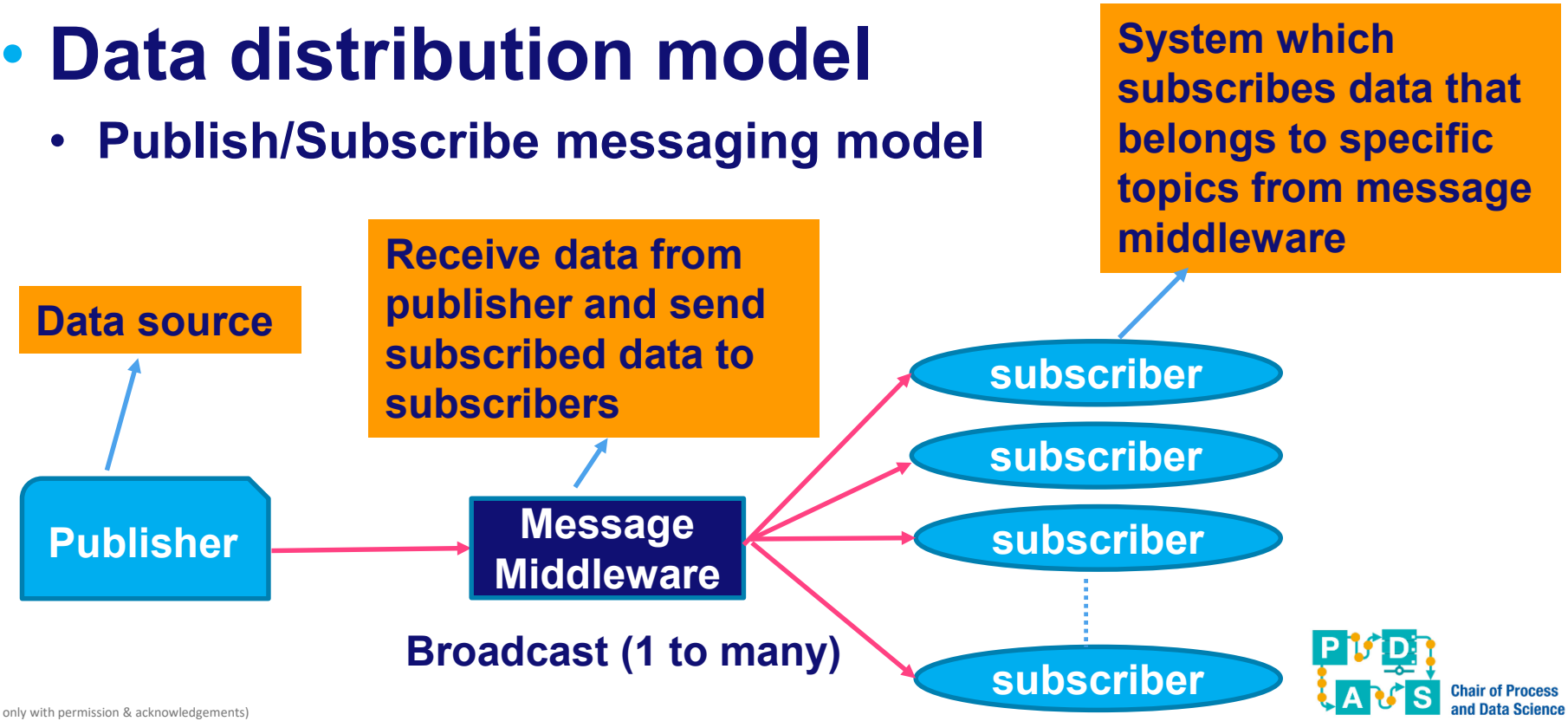


Unicast

Chair of Process and Data Science

Streaming

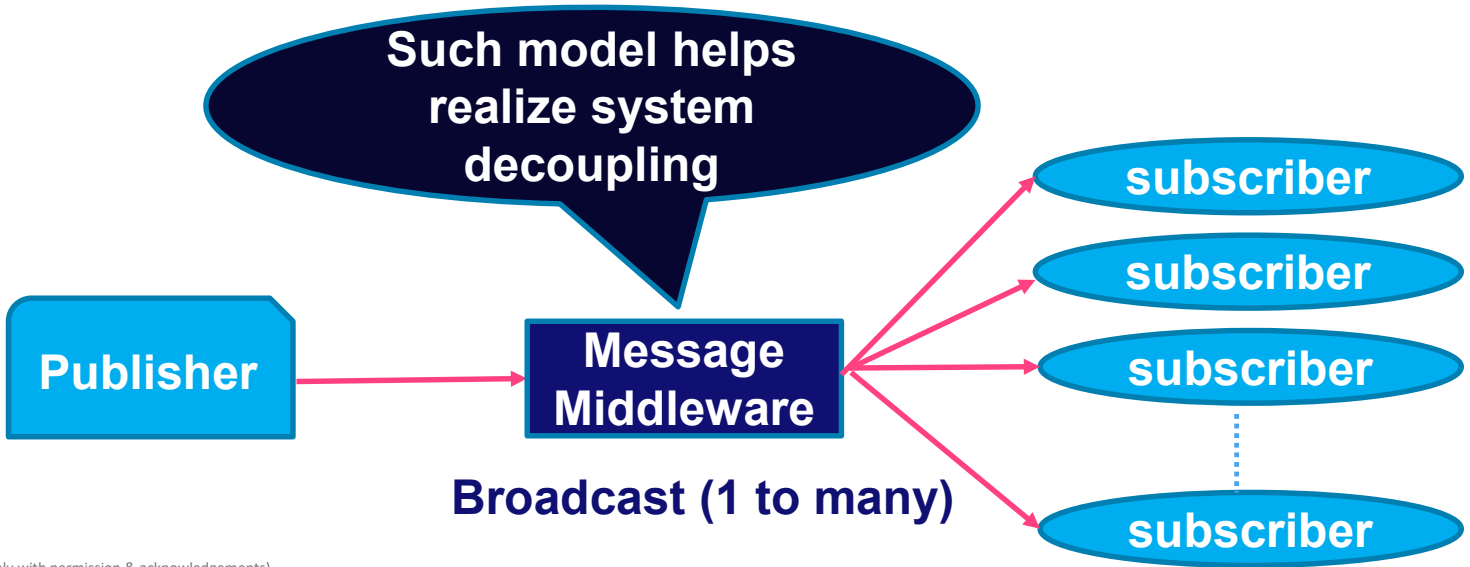
- **Data distribution model**
  - **Publish/Subscribe messaging model**



Chair of Process and Data Science

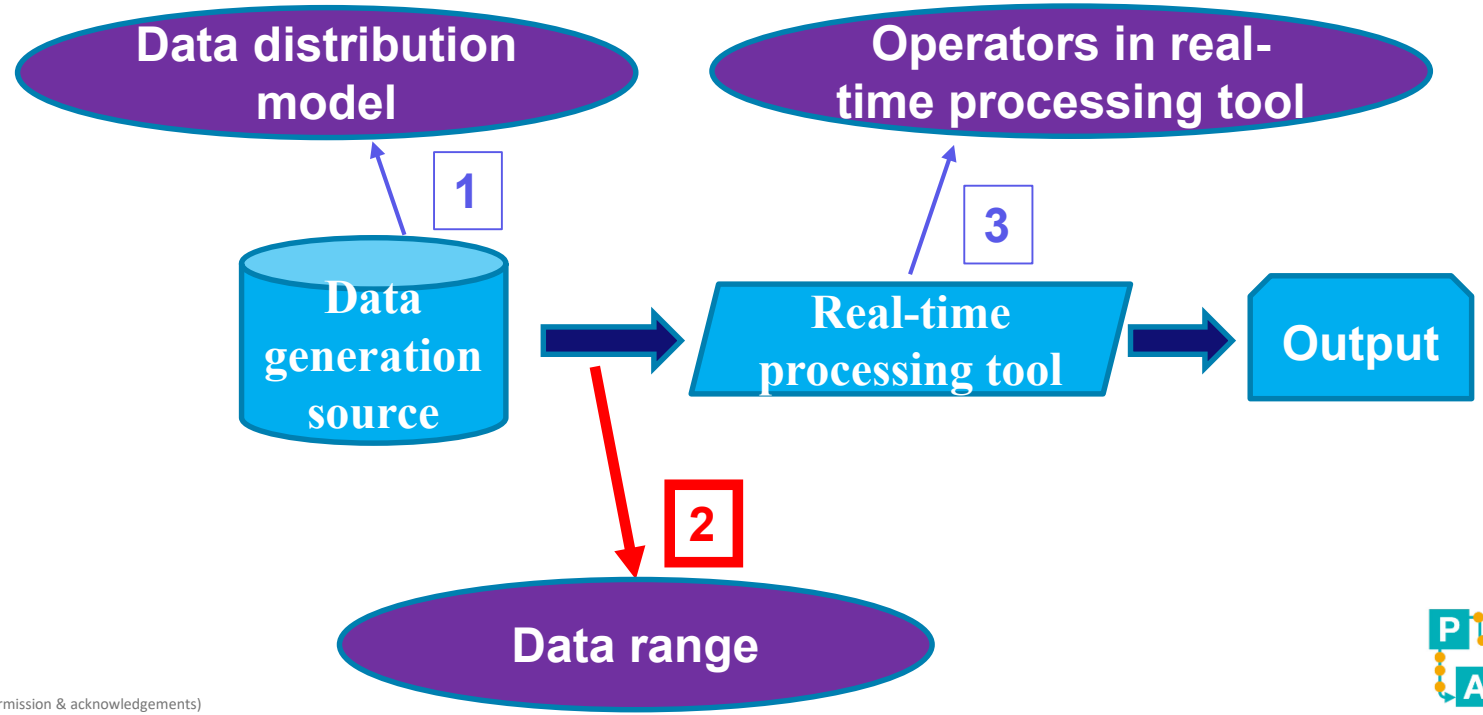
Streaming

- **Data distribution model**
  - **Publish/Subscribe messaging model**



Streaming

- **Important elements in streaming**



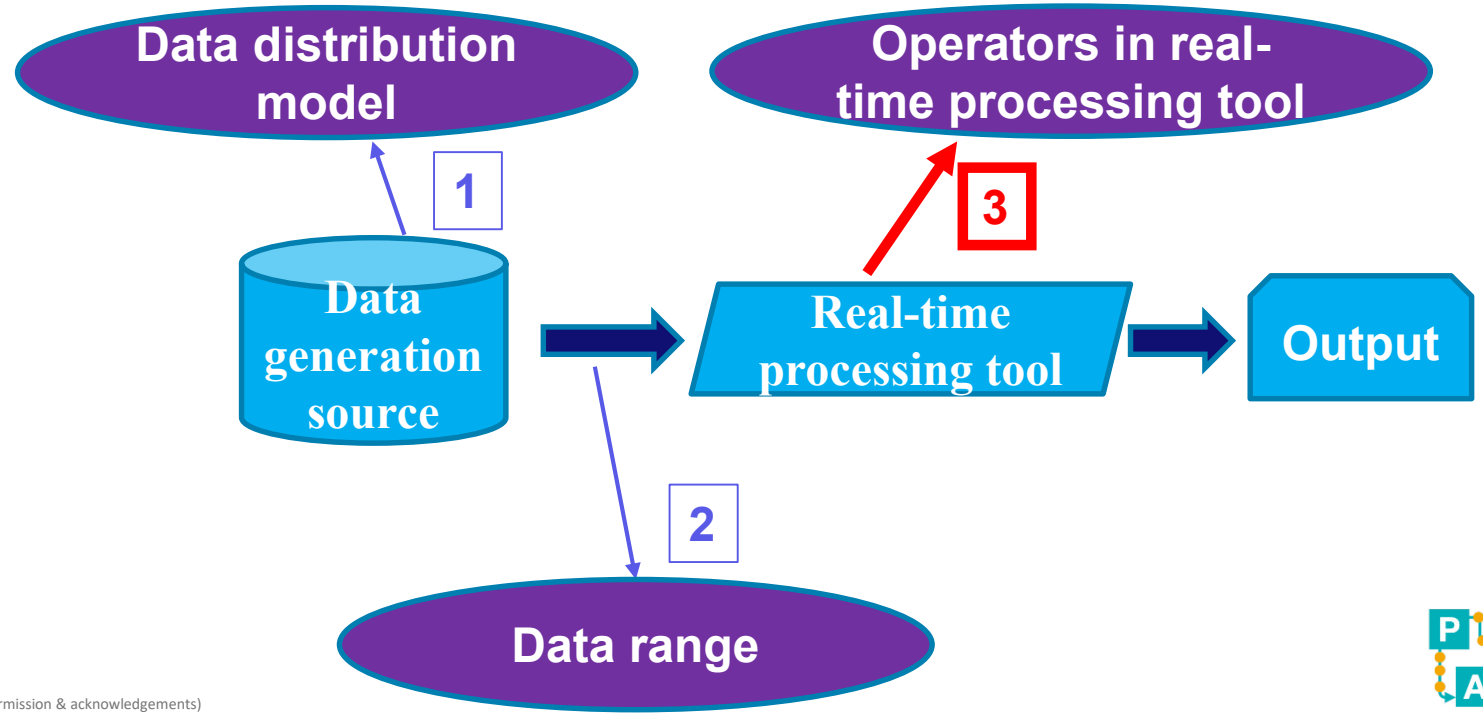
Chair of Process and Data Science

Data range

- What data to consider?
  - Data stream model: Consider all the data objects from a specific time point until now (e.g., for calculating the average speed of a car, consider all its speed data received until now).
  - Sliding window model: compute over a fixed size of window in the stream (e.g., compute the average speed of a car, only consider the speed data received during the last 10 days).
- Due to capacity constraints not everything can be stored or processed!

Streaming

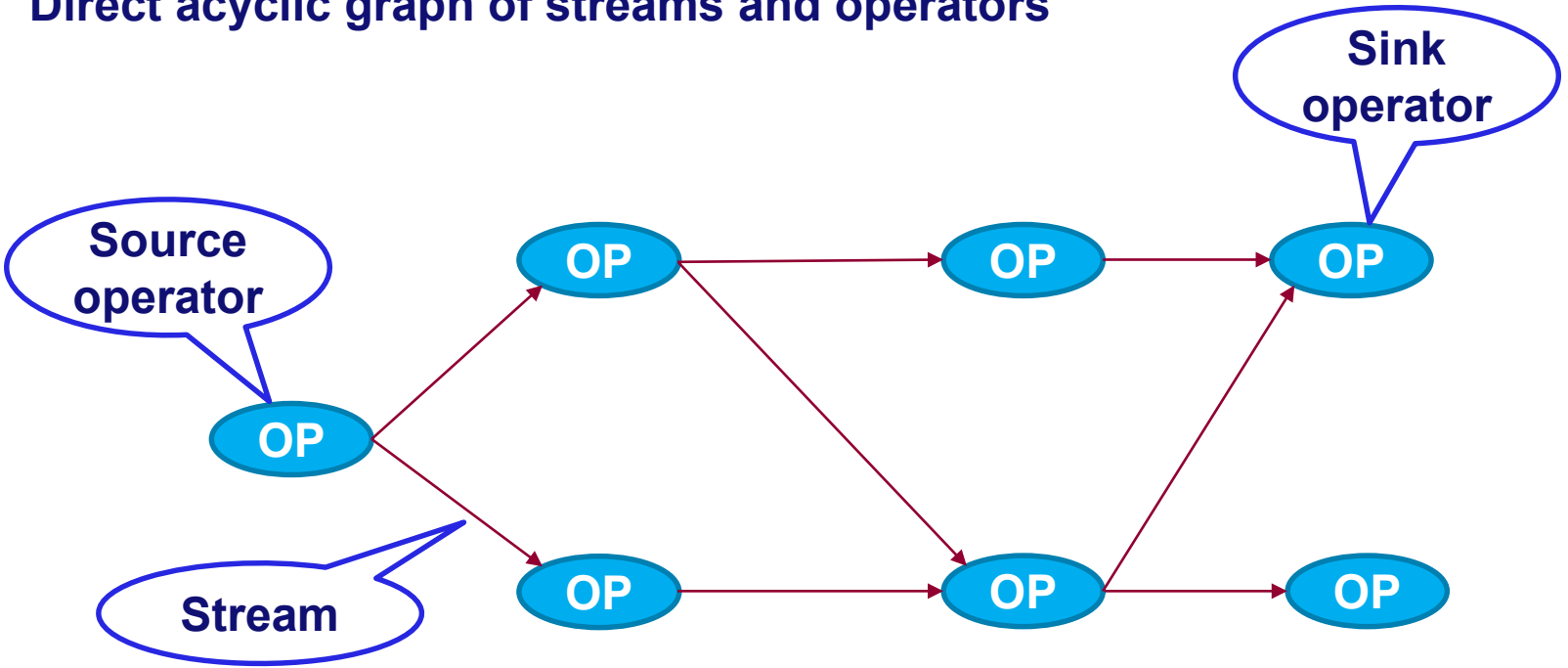
- **Important elements in streaming**



Chair of Process and Data Science

Operator in real-time processing

Direct acyclic graph of streams and operators

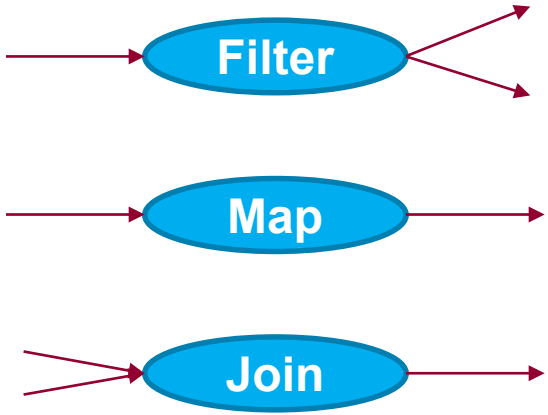


Two types of operators in real-time processing

- **Stateless operators:**
  - **Don’t record historical data object in memory**
  - **One-by-one processing possible**
- **Stateful operators:**
  - **Need to record historical data object in memory**
  - **Outputs depend on multiple input data objects**

Streaming

- **Stateless operators**



Filter data objects according to one or more conditions.

Convert each data object.

Merge multiple streams into one stream.

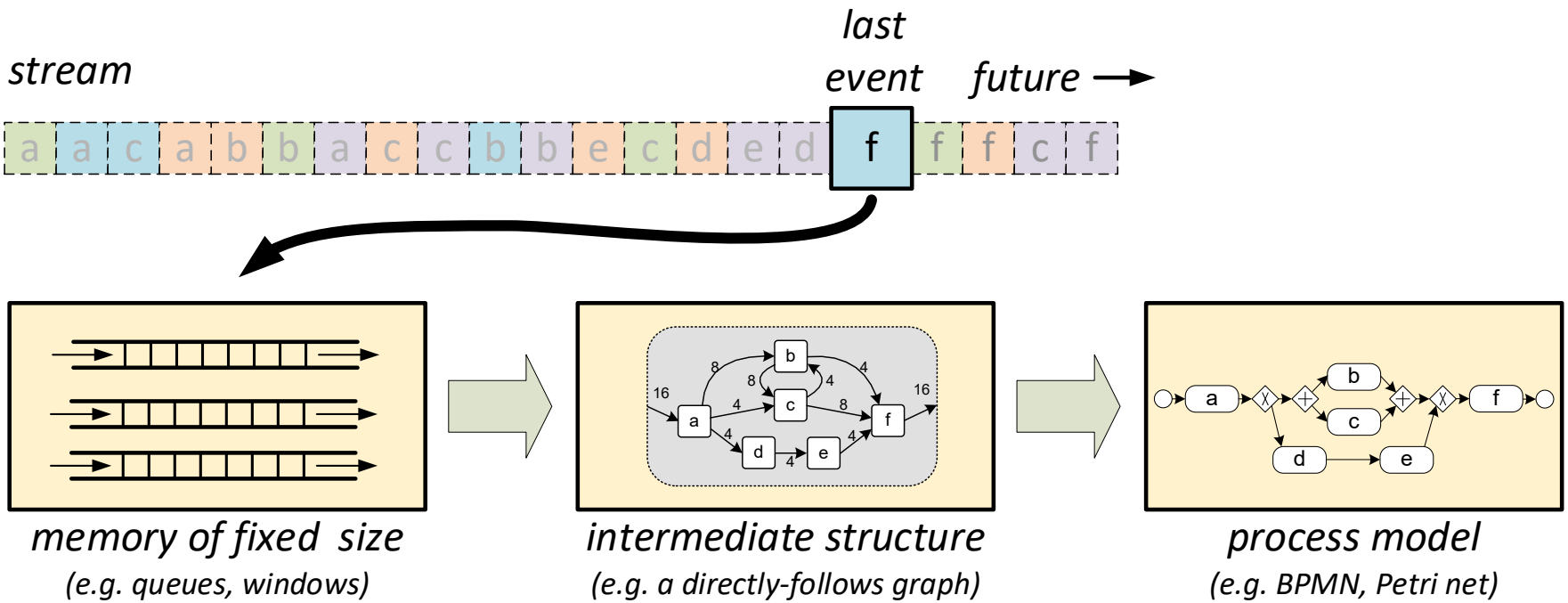
Streaming

- **Stateful operators**

Aggregate multiple data objects.
(e.g., compute average speed of data objects in last five minutes)

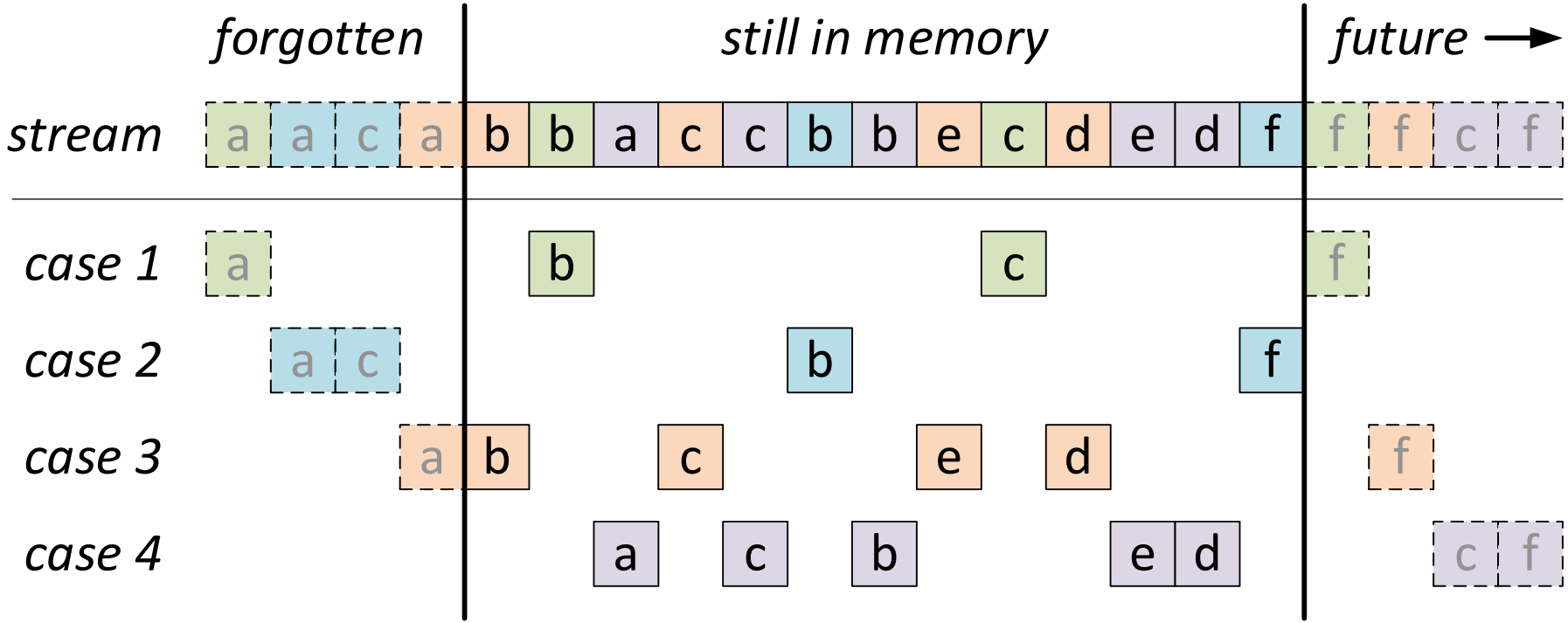
Join data objects from two streams given a specific predicate.
(e.g., for the last 3 data objects from each stream, join every pair with the same ID)

Example: Streaming process mining

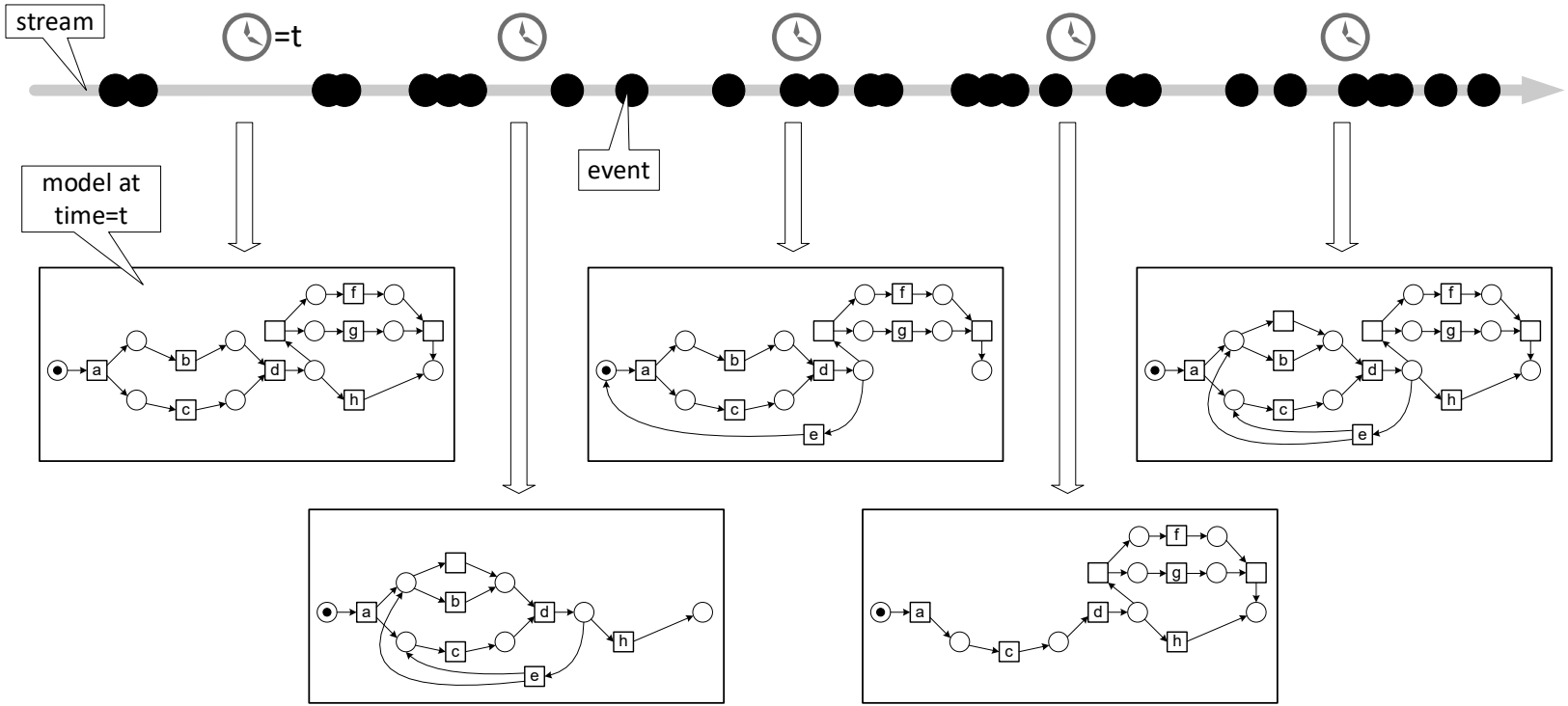


Chair of Process and Data Science

Example: Streaming process mining



Example: Streaming process mining



Relates to concept drift (see later).

Chair of Process and Data Science

Streaming k-means

Initial non-streaming k-means algorithm

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

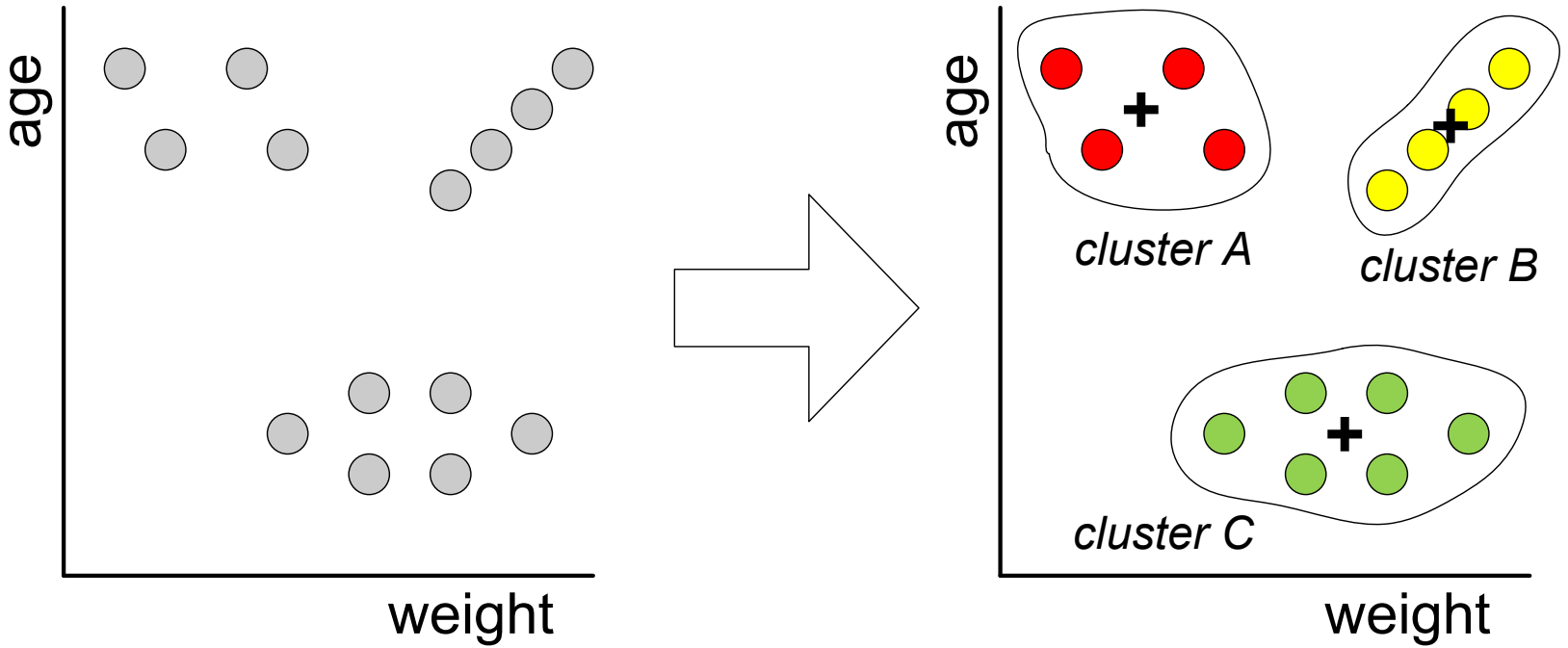
- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

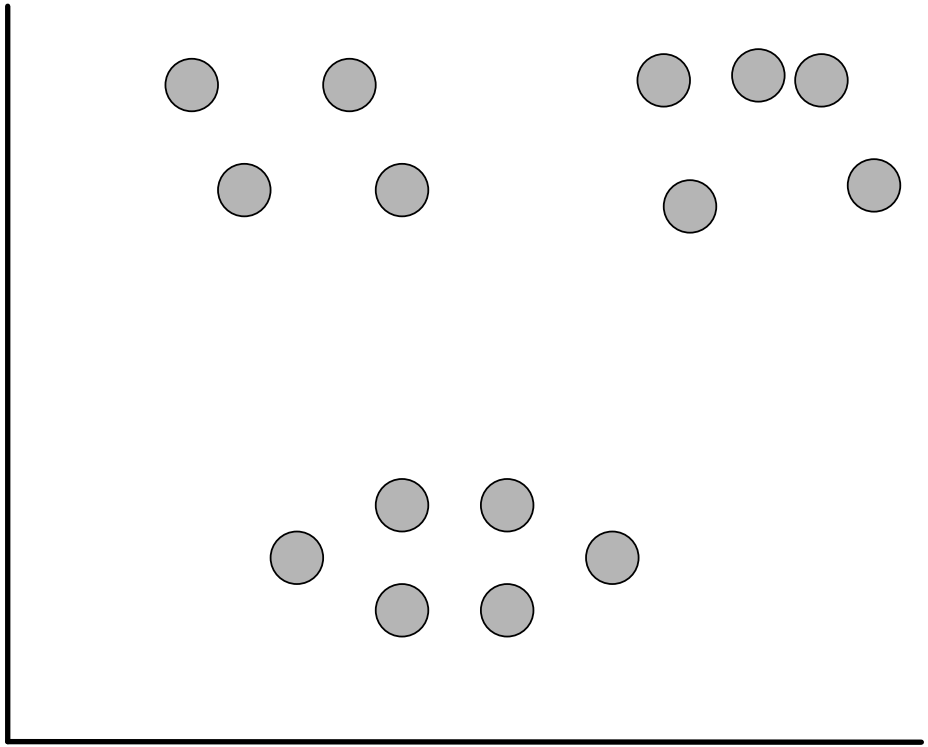
Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

Clustering



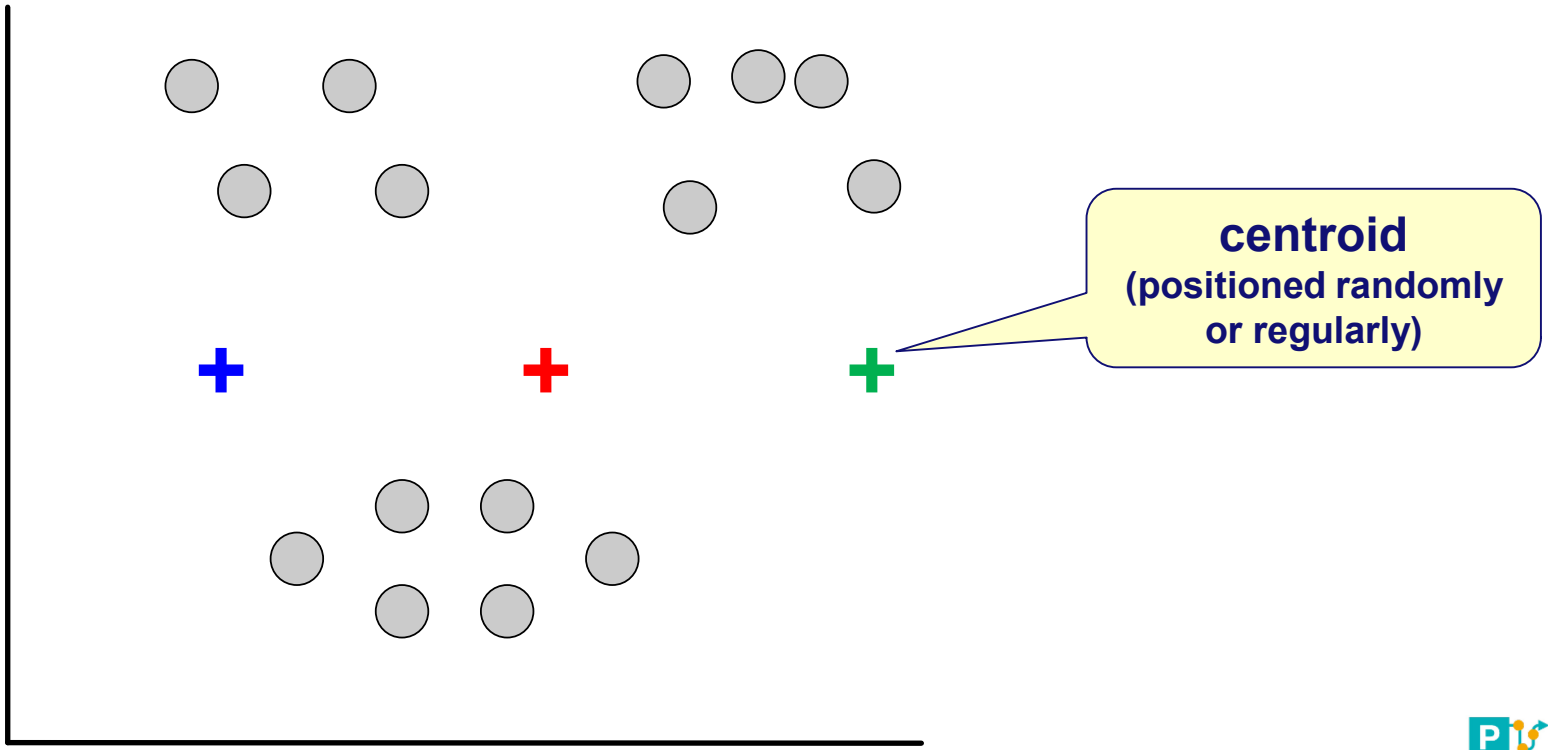
k-means clustering



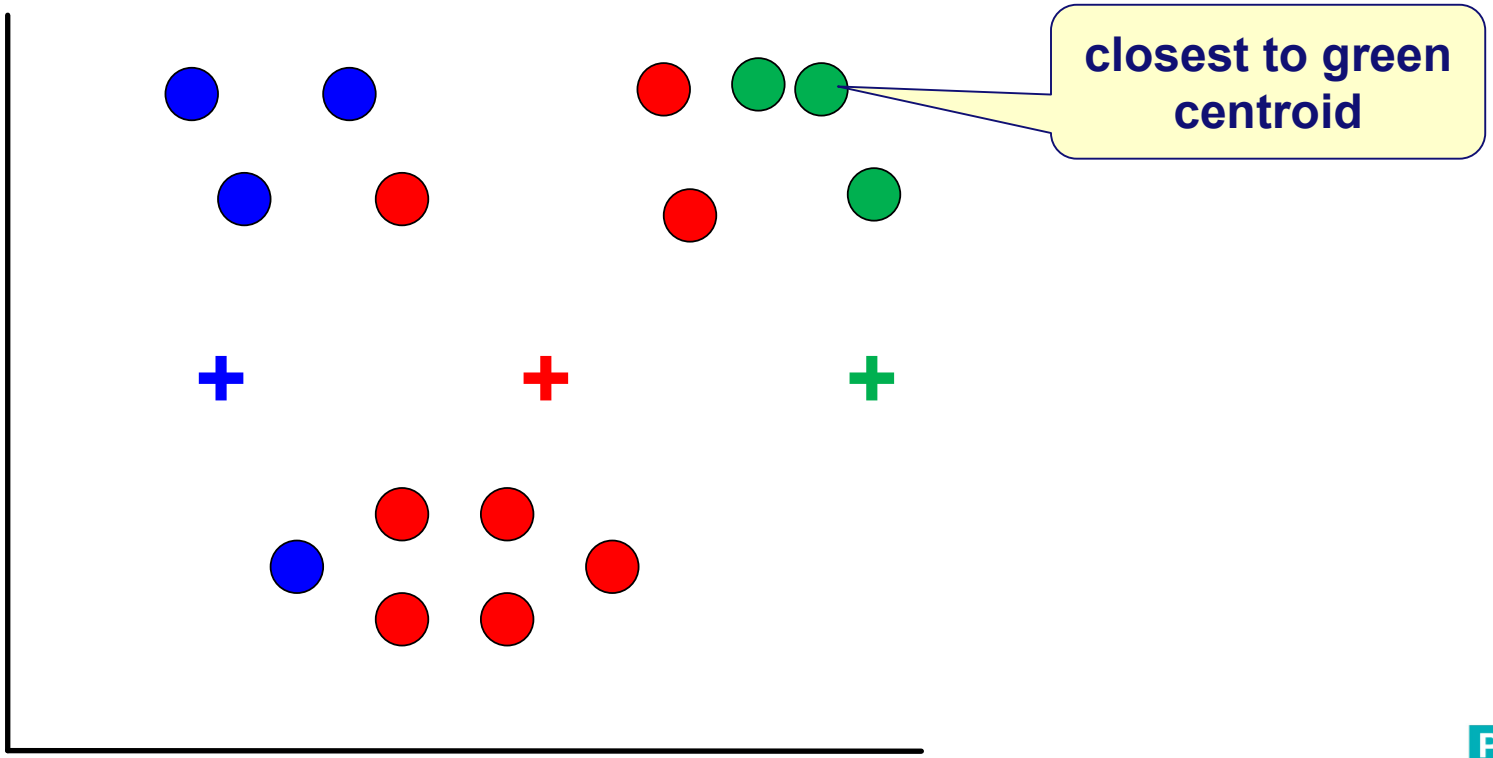
assume $k=3$

Just an illustration.
May be misleading:
often many dimensions!

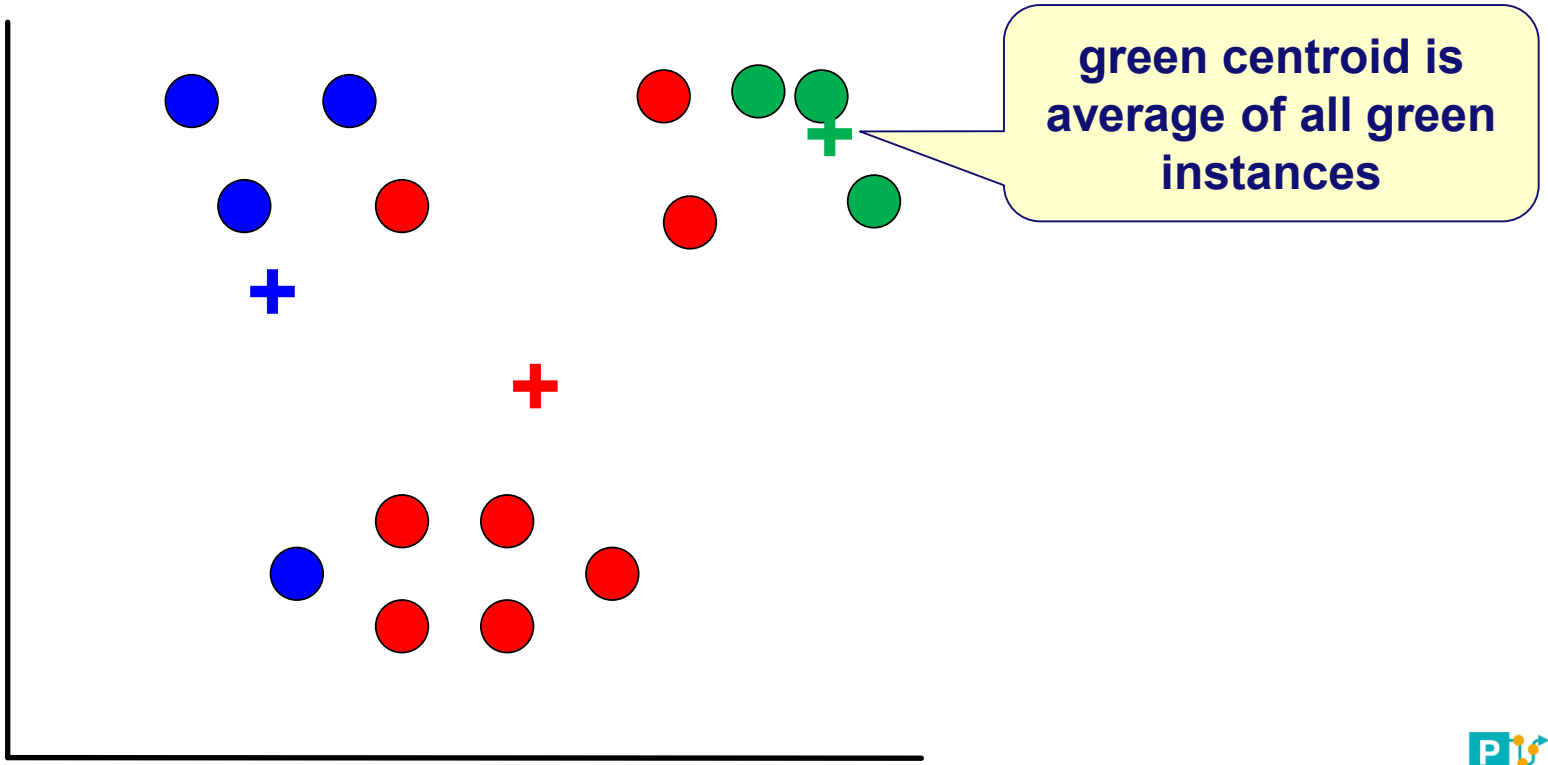
Generate *k*=3 centroids (e.g., random)



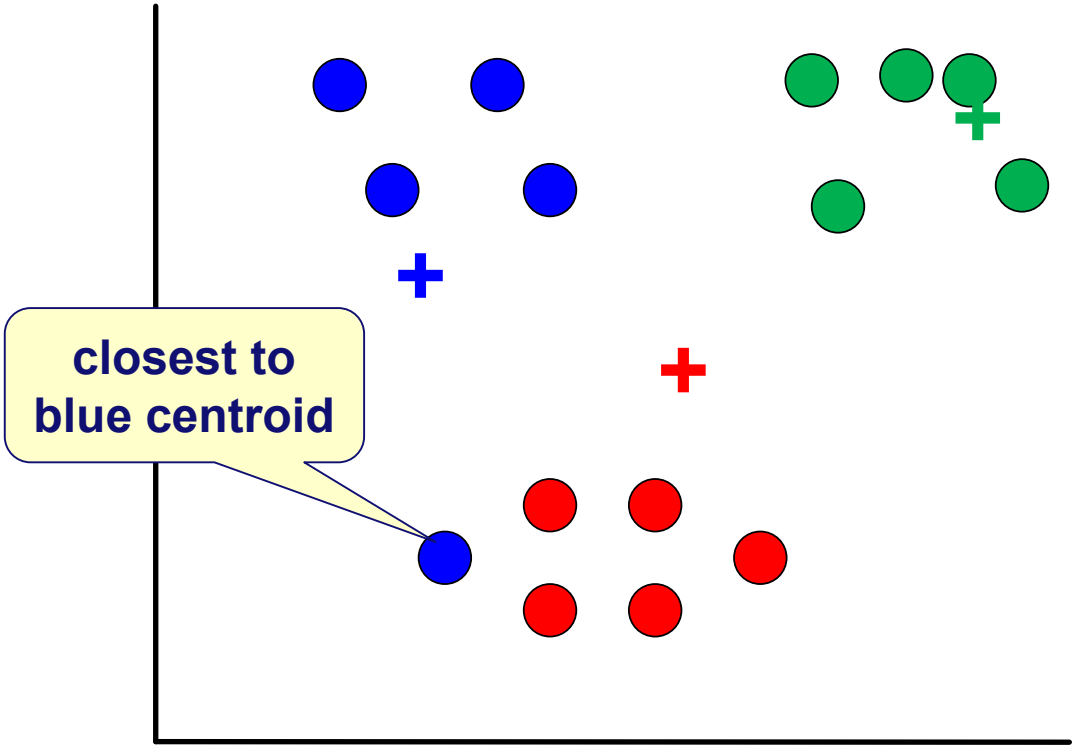
Assign instances to closest centroid



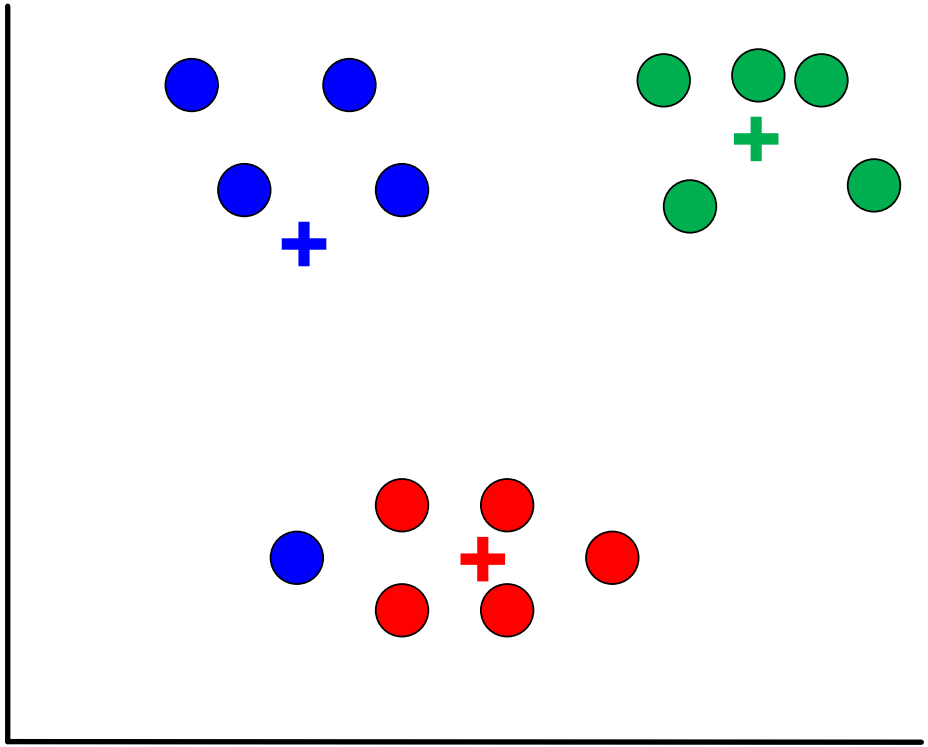
Recompute centroids based on assigned instances



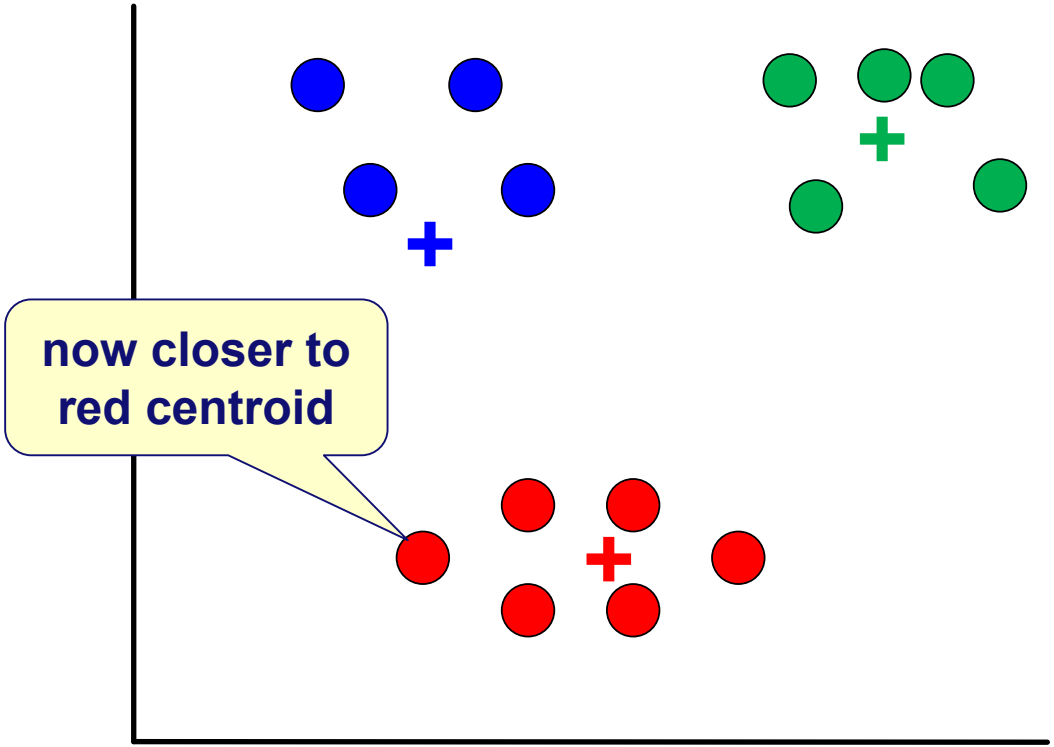
Assign instances to closest centroid



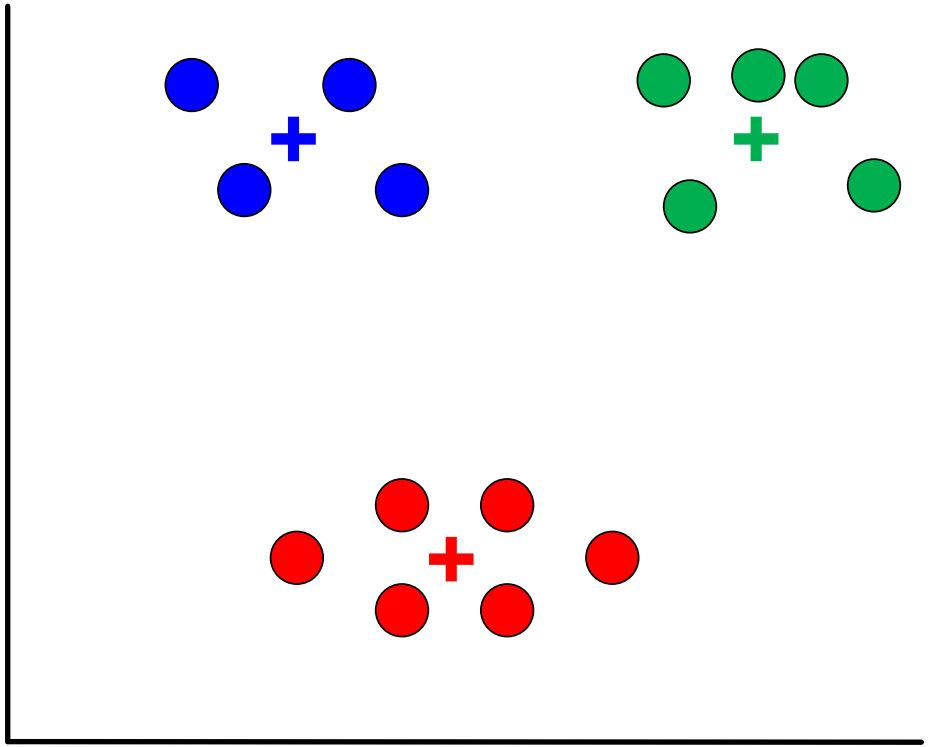
Recompute centroids based on assigned instances



Assign instances to closest centroid



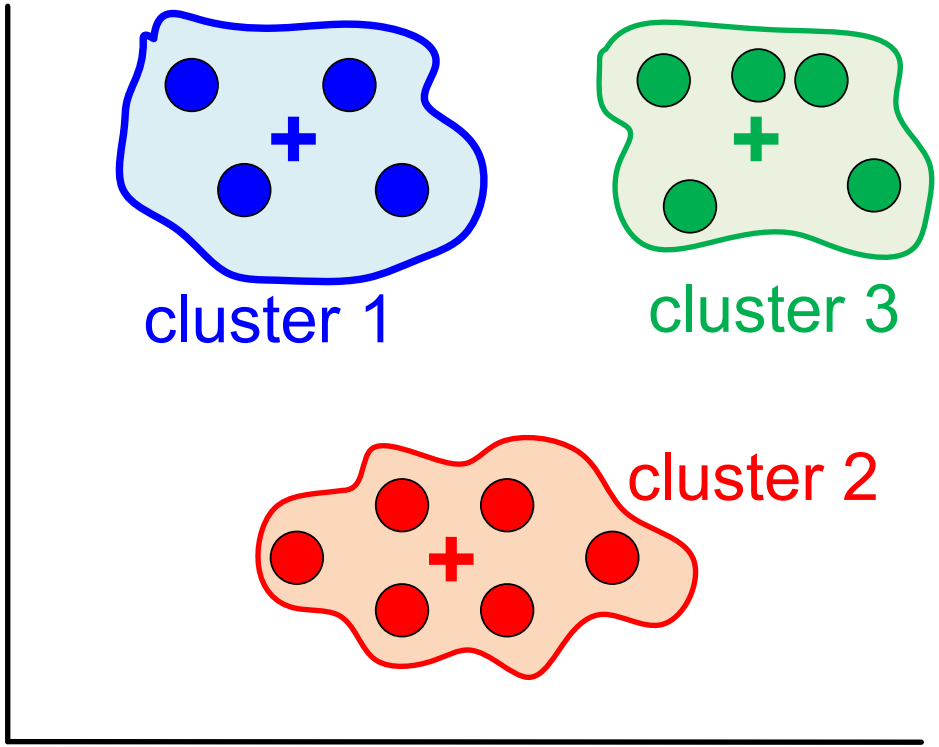
Recompute centroids based on assigned instances



Fixpoint has been reached: Nothing changes anymore.

Due to non-deterministic nature (random initial centroids), experiment may be repeated multiple times. Select "best clustering" at end.

Clusters returned



Main idea:
The instances in a cluster are more similar to each other than to those in other clusters.

Chair of Process and Data Science


Assume now that we have limited
storage and are too slow to access old
events and do computations over them!

Streaming k-means algorithm

- Assume we cannot store all instances.
- New instances arrive in batches and are processed in batch (and then forgotten).
- Behavior is expected to change over time (concept drift). The clusters should reflect both past and current instances.
- Many variations are possible, we just describe one variant.

Some notation (1/2)

- At any point in time there are k clusters $\{1, 2, \dots, k\}$.
- c_i^t is the centroid of cluster $i \in \{1, 2, \dots, k\}$ before the t-th batch B^t arrives.
- Hence, $\{c_1^t, c_2^t, \dots, c_k^t\}$ are the centroids at time t.
- $\{c_1^1, c_2^1, \dots, c_k^1\}$ are the initial centroids.

Some notation (2/2)

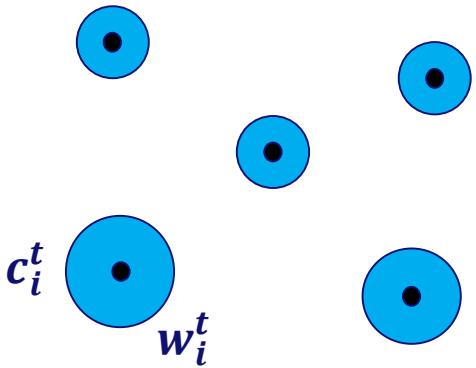
- At any point in time there are k clusters $\{1, 2, \dots, k\}$.
- w_i^t is the weight of cluster i before the t-th batch B^t arrives.
- Hence, $\{w_1^t, w_2^t, \dots, w_k^t\}$ are the weights at time t.
- $\{w_1^1, w_2^1, \dots, w_k^1\}$ are the initial weights.

Main idea

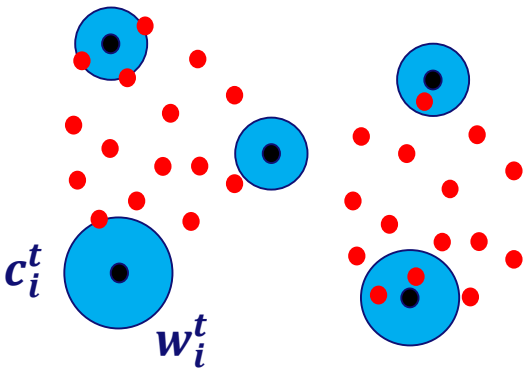
- For each new batch arrival B^t:
  - Assign each instance to the nearest cluster centroid.
  - Update the cluster centroids and weights.
- If the weight of a cluster i drops below a preset threshold δ, then cluster i is removed and the largest cluster j is split into two clusters, one taking over the role of i.
- I.e., the old cluster j is split into the new clusters i and j.

Example: Update clusters

clusters at time t



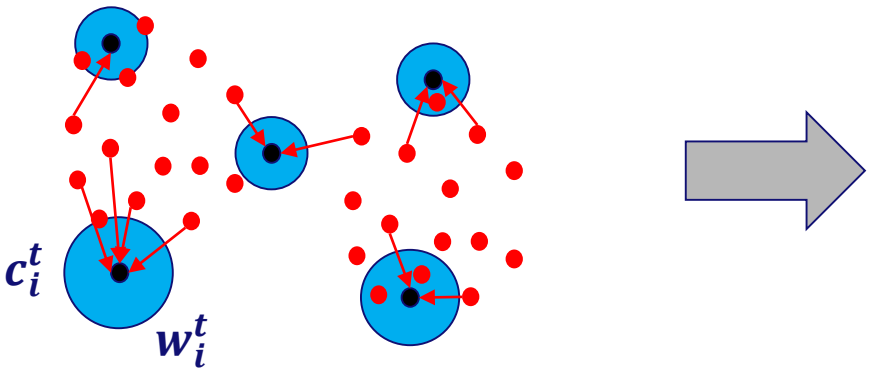
new batch



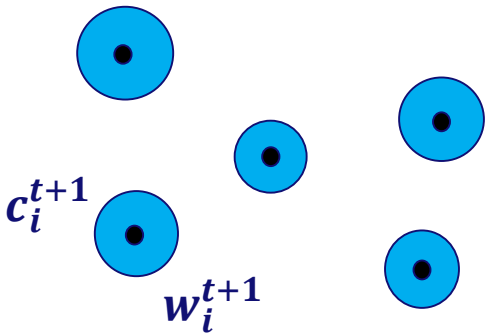
Chair of Process and Data Science

Example: Update clusters

clusters at time t and new batch

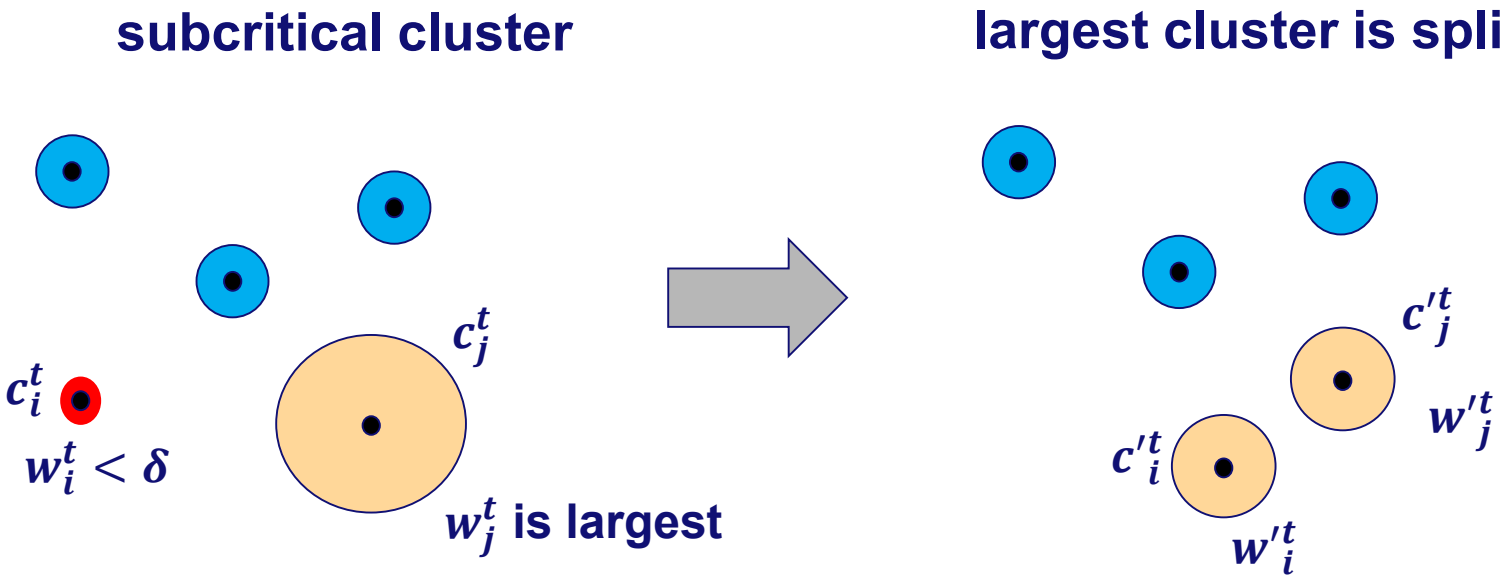


clusters at time $t+1$



Chair of Process and Data Science

Example: Splitting



Different ways of splitting c_j^t (outside of scope).

Chair of Process and Data Science

Batch arrival $B^t = B_1^t \cup B_2^t, \ldots, B_k^t$

- $\{c_1^t, c_2^t, \ldots, c_k^t\}$ and $\{w_1^t, w_2^t, \ldots, w_k^t\}$ describe the centroids at time t when a batch B^t of instances arrives.
- $B^t = B_1^t \cup B_2^t, \ldots, B_k^t$ are the instances assigned to the nearest centroid.
- $\{bc_1^t, bc_2^t, \ldots, bc_k^t\}$ and $\{bw_1^t, bw_2^t, \ldots, bw_k^t\}$ are the centroids and weights of the partitioned batches where $bw_i^t = |B_i^t|$ and bc_i^t is computed as usual (average vector).

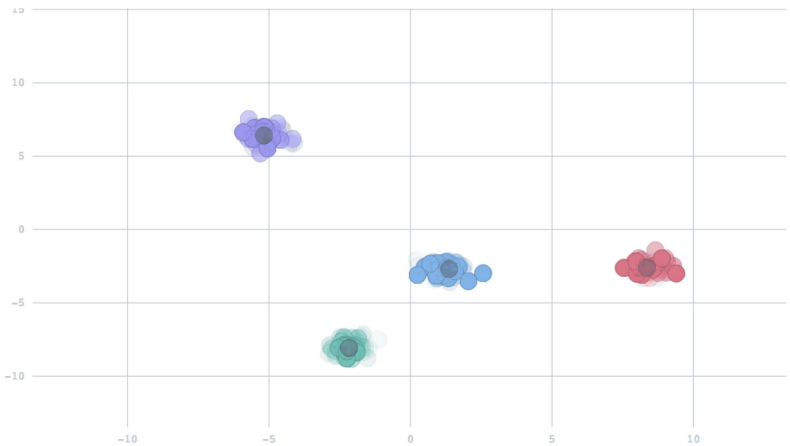
Decay factor $\alpha \in [0, 1]$

- $\alpha \in [0, 1]$ is the decay factor. The smaller α, the bigger the decay.
- $c_i^{t+1} = \frac{\alpha \cdot w_i^t \cdot c_i^t + bc_i^t}{\alpha \cdot w_i^t + bw_i^t}$ is the new centroid after adding B_i^t.

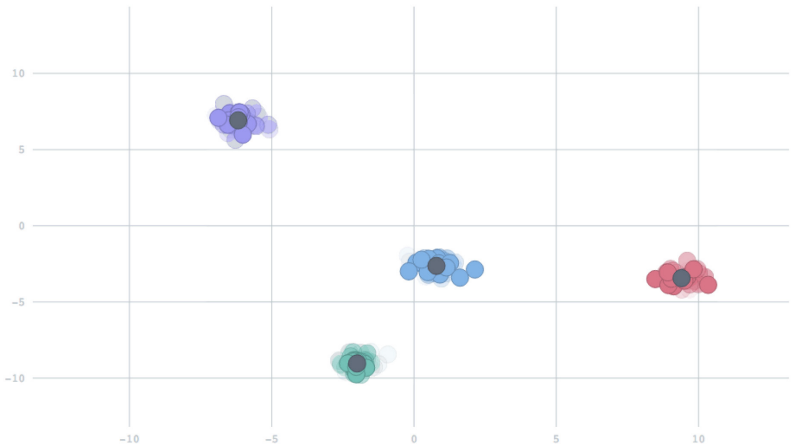
  Note that c_i^t and bc_i^t are vectors.
- $w_i^{t+1} = \alpha \cdot w_i^t + bw_i^t$ is the new centroid after adding B_i^t.

- Remove cluster i if $w_i^{t+1} < \delta$ and split the largest cluster into two.

The effect of the decay factor



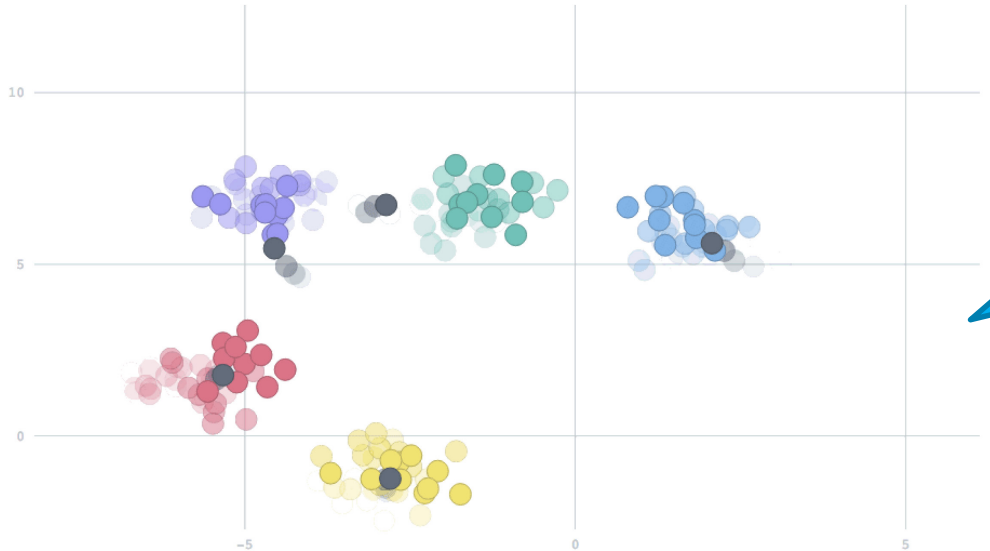
Higher value of α
(slower adjustment)



Lower value of α
(faster adjustment)

https://databricks.com/blog/2015/01/28/introducing-streaming-k-means-in-spark-1-2.html

The effect of the decay factor



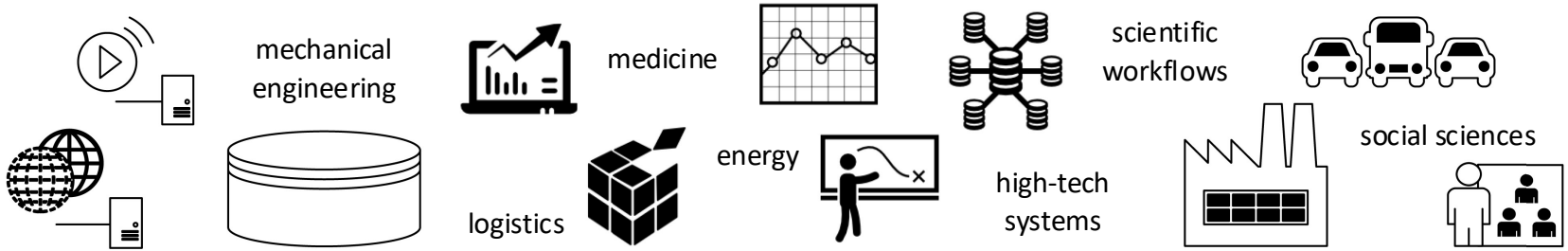
Appropriate setting of decay factor make centers change smoothly

https://databricks.com/blog/2015/01/28/introducing-streaming-k-means-in-spark-1-2.html

Concept drift

- Different types of drift:
  - Sudden drifts (e.g., due to Brexit).
  - Gradual drifts (e.g., due to improving economy).
  - Seasonal drifts (e.g., due to weather).
  - Intermitted drifts (e.g., due to system failures).
- What should the model describe?
  - Recent behaviors?
  - Long-term behaviors?

Conclusion



infrastructure

"volume and velocity"

- big data infrastructures
- distributed systems
- data engineering
- programming
- security
- ...

analysis

"extracting knowledge"

- statistics
- data/process mining
- machine learning
- artificial intelligence
- visualization
- ...

effect

"people, organizations, society"

- ethics & privacy
- IT law
- operations management
- business models
- entrepreneurship
- ...

Chair of Process and Data Science

Summary

- Big Data: Relevance and Applications
- Characteristics of Big Data: The 4 V's
- Big Data Infrastructures
- MapReduce
- Streaming
- Streaming k-means

Relevant Literature

- Viktor Mazer-Schönberger. “Big Data: A Revolution That Will Transform How We Live, Work, and Think”. 04.03.2014.
- Apache Hadoop: MapReduce Tutorial. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
- Streaming 101: The world beyond batch. https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101
- Streaming 102: https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-102
- van der Aalst, Wil M.P. “Process Mining in the Large: A Tutorial”. Business Intelligence, pp 33-76.
- van der Aalst, Wil M.P. “Decomposing Petri nets for process mining: A generic approach”. Distributed and Parallel Databases, Volume 31, pp 471-507.
- Nada Elgendz, Ahmed Elragal. “Big Data Analytics: A Literature Review Paper”. Industrial Conference on Data Mining. pp 214-227.
- Tom White. “Hadoop: The Definitive Guide”. O’Reillz Media, April 2015.

Instruction on Friday

- It is essential that you have Hadoop installed before Friday. It does not make sense to try an install it during the instruction. See posting Yaguang Sun.
- Tomorrow (Thursday lecture), Yaguang Sun will demo this, summarize the installation, and help people that did not succeed.
- During the instruction you will learn how to implement a MapReduce algorithm and run your code on Hadoop.

#	Lecture	date	day
Lecture 1	Introduction	10/10/2018	Wednesday
Lecture 2	Crash Course in		
Instruction 1	*Python*		
Lecture 3	Basic data visua		
Lecture 4	Decision trees		
Instruction 2	*Decision trees*		
Lecture 5	Regression		
Lecture 6	Support vector		
Instruction 3	*Regression and*		
Lecture 7	Neural network		
Instruction 4	*Neural network*		
Lecture 8	Neural network		
Lecture 9	Evaluation of su		
Instruction 5	*Neural network*		
Lecture 10	Clustering		
Lecture 11	Frequent items		
Lecture 12	Association rules	21/11/2018	Wednesday
Lecture 13	Sequence mining	22/11/2018	Thursday
Instruction 6	*Clustering, frequent items sets, association rules*	*23/11/201*	*riday*
Lecture 14	Process mining (unsupervised)	28/11/2	Wednesday
Lecture 15	Process mining (supervised)	29/11	Thursday
Instruction 7	*Process mining and sequence mining*	*30/*	*Friday*
Lecture 16	Text mining (1/2)	0	18 Wednesday
Instruction 8	*Text mining and process mining*		*18 Thursday !!*
Lecture 17	Text mining (2/2)		018 Wednesday
Lecture 18	Data preprocessing, data quality, binning, etc.		2018 Thursday
Lecture 19	Visual analytics & information visualization		/2018 Wednesday
backup			2/2018 Thursday
Instruction 9	*Text*		*riday*
Lecture 20	Res		ednesday
Lecture 21	Res		ursday
Instruction 10	*Res*		*iday*
Lecture 22	Big		ednesday
Lecture 23	Big		ursday
Instruction 11	*Big*		*iday*
Lecture 24	Clos		ednesday
backup		24/01/2019	Thursday
Instruction 12	*Example exam questions*	*25/01/2018*	*Friday*
backup		30/01/2019	Wednesday
backup		31/01/2019	Thursday
extra	*Question hour*	*01/02/2019*	*Friday*

Lecture 22	Big data (1/2)	16/01/2019	Wednesday
Lecture 23	Big data (2/2)	17/01/2019	Thursday
Instruction 11	*Big data*	*18/01/2019*	*Friday*
Lecture 24	Closing	23/01/2019	Wednesday
backup		24/01/2019	Thursday
Instruction 12	*Example exam questions*	*25/01/2018*	*Friday*
backup		30/01/2019	Wednesday
backup		31/01/2019	Thursday
extra	*Question hour*	*01/02/2019*	*Friday*

Solutions for two trail exams are given.

You can ask questions to the instructors and get two trail exams

#	Lecture	date	day
Lecture 1	Introduction	10/10/2018	Wednesday
Lecture 2	Crash Course in		
Instruction 1	*Python*		
Lecture 3	Basic data visua		
Lecture 4	Decision trees		
Instruction 2	*Decision trees*		
Lecture 5	Regression		
Lecture 6	Support vector		
Instruction 3	*Regression and*		
Lecture 7	Neural network		
Instruction 4	*Neural network*		
Lecture 8	Neural network		
Lecture 9	Evaluation of su		
Instruction 5	*Neural network*		
Lecture 10	Clustering		
Lecture 11	Frequent items		
Lecture 12	Association rules	21/11/2018	day
Lecture 13	Sequence mining	22/11/201	
Instruction 6	*Clustering, frequent items sets, association rules*	*23/11/201*	
Lecture 14	Process mining (unsupervised)	28/11/20	y
Lecture 15	Process mining (supervised)	29/11/20	
Instruction 7	*Process mining and sequence mining*	*30/11/20*	
Lecture 16	Text mining (1/2)	05/12/2	
Instruction 8	*Text mining and process mining*	*06/12/2*	
Lecture 17	Text mining (2/2)	12/12/	
Lecture 18	Data preprocessing, data quality, binning, etc.		
Lecture 19	Visual analytics & information visualization		
backup			
Instruction 9	*Text mining, preprocessing and visualization*		
Lecture 20	Responsible data science (1/2)		
Lecture 21	Responsible data science (2/2)		
Instruction 10	*Responsible data science*		
Lecture 22	Big data (1/2)		
Lecture 23	Big data (2/2)		
Instruction 11	*Big data*		
Lecture 24	Closing	23/01/2019	Wednesday
backup		24/01/2019	Thursday
Instruction 12	*Example exam questions*	*25/01/2018*	*Friday*
backup		30/01/2019	Wednesday
backup		31/01/2019	Thursday
extra	*Question hour*	*01/02/2019*	*Friday*

Lecture 22	Big data (1/2)	16/01/2019	Wednesday
Lecture 23	Big data (2/2)	17/01/2019	Thursday
Instruction 11	*Big data*	*18/01/2019*	*Friday*
Lecture 24	Closing	23/01/2019	Wednesday
backup		24/01/2019	Thursday
Instruction 12	*Example exam questions*	*25/01/2018*	*Friday*
backup		30/01/2019	Wednesday
backup		31/01/2019	Thursday
extra	*Question hour*	*01/02/2019*	*Friday*

Prepare questions and send them before Monday 21/1/2019 12.00.

Yaguang Sun provides support on 17/1/2019 for those that could not install Hadoop. Instruction assumes Hadoop is up and running (no regular lecture)