DATA CENTRIC SCIENCES

Big Data Analytics and Management

Genoveva Vargas-Solar

Senior Scientist, French Council of Scientific Research

Javier A. Espinosa Oviedo

Postdoctoral fellow, Barcelona Super Computing Centre

Verano Académico, Télécom SudParis, 26th June 2017

http://vargas-solar.com/datacentric-sciences/

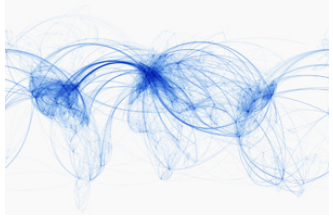
WHAT ARE DATA CENTRIC SCIENCES ?

THE STUDY OF COMPLEX SYSTEMS

Social Data Science

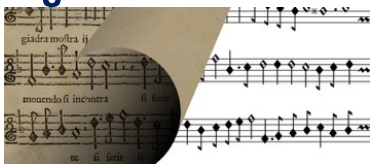
Data Science

Network Science



Computational Science

Digital humanities

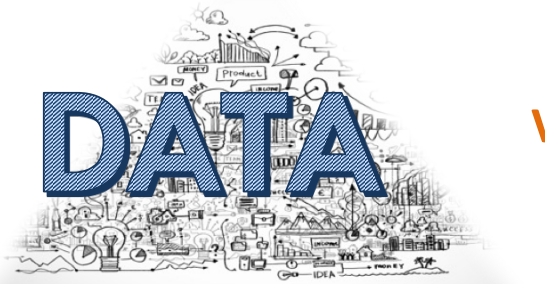


Computation
(Algorithm: mathematical model)

Experiment
(Architecture: computing environment)

Volume

Velocity

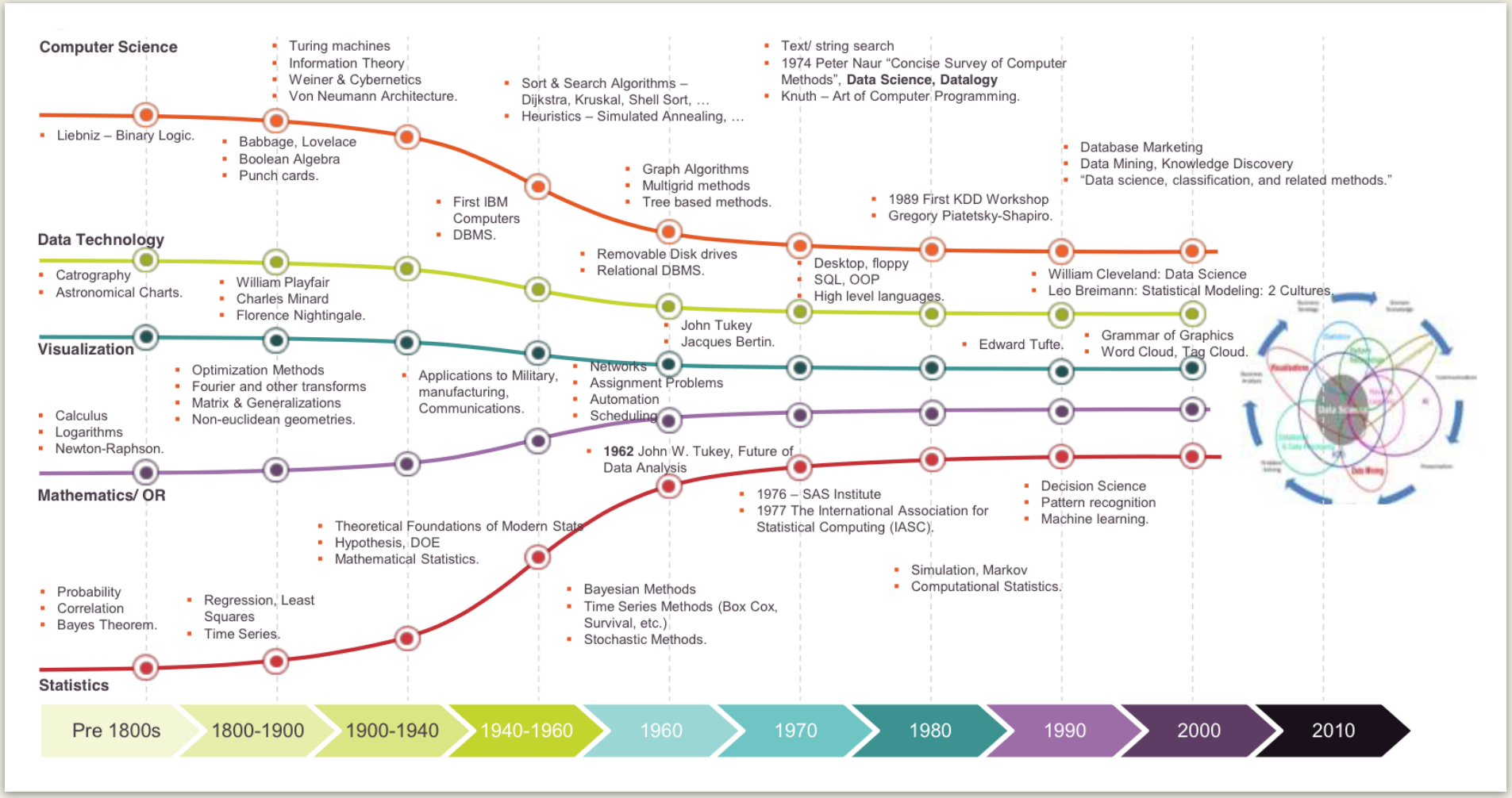


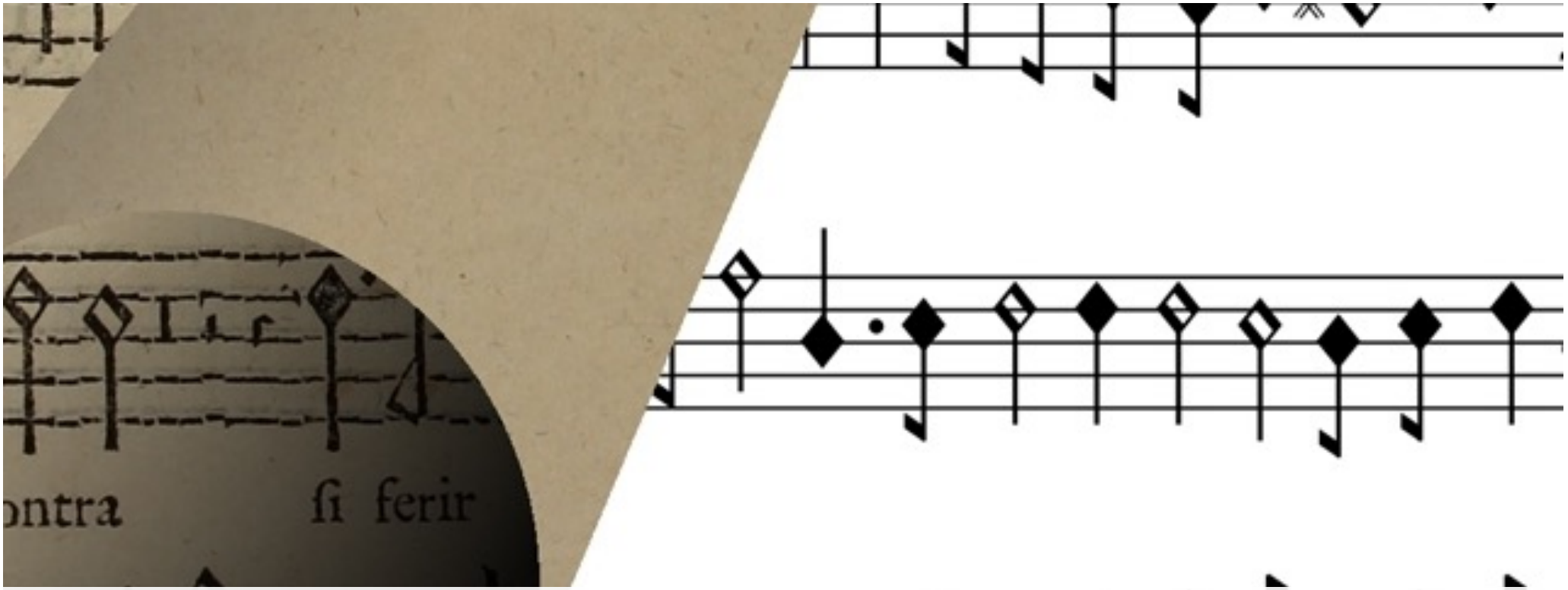
Variety

Value

Veracity

1000 Yottabytes	1 Brontobyte
1000 Brontobytes	1 Geopbyte

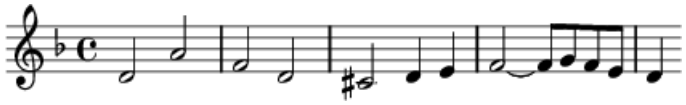

Computer Science
Liebniz – Binary Logic.
Turing machines
Information Theory
Weiner & Cybernetics
Von Neumann Architecture.
Babbage, Lovelace
Boolean Algebra
Punch cards.
Sort & Search Algorithms – Dijkstra, Kruskal, Shell Sort, …
Heuristics – Simulated Annealing, …
First IBM Computers
DBMS.
Graph Algorithms
Multigrid methods
Tree based methods.
Text/ string search
1974 Peter Naur "Concise Survey of Computer Methods", Data Science, Datalogy
Knuth – Art of Computer Programming.
1989 First KDD Workshop
Gregory Piatetsky-Shapiro.
Database Marketing
Data Mining, Knowledge Discovery
"Data science, classification, and related methods."
Data Technology
Catrography
Astronomical Charts.
William Playfair
Charles Minard
Florence Nightingale.
Removable Disk drives
Relational DBMS.
Desktop, floppy
SQL, OOP
High level languages.
William Cleveland: Data Science
Leo Breimann: Statistical Modeling: 2 Cultures.
Visualization
John Tukey
Jacques Bertin.
Edward Tufte.
Grammar of Graphics
Word Cloud, Tag Cloud.
Optimization Methods
Fourier and other transforms
Matrix & Generalizations
Non-euclidean geometries.
Applications to Military, manufacturing, Communications.
Networks
Assignment Problems
Automation
Scheduling
Calculus
Logarithms
Newton-Raphson.
Mathematics/ OR
1962 John W. Tukey, Future of Data Analysis
1976 – SAS Institute
1977 The International Association for Statistical Computing (IASC).
Decision Science
Pattern recognition
Machine learning.
Theoretical Foundations of Modern Stats
Hypothesis, DOE
Mathematical Statistics.
Simulation, Markov
Computational Statistics.
Probability
Correlation
Bayes Theorem.
Regression, Least Squares
Time Series.
Bayesian Methods
Time Series Methods (Box Cox, Survival, etc.)
Stochastic Methods.
Statistics
Pre 1800s
1800-1900
1900-1940
1940-1960
1960
1970
1980
1990
2000
2010



DIGITAL HUMANITIES

What makes Bach sound like Bach?

The composer Johann Sebastian Bach left behind an incomplete fugue upon his death, either as an unfinished work or perhaps as a puzzle for future composers to solve



The Art of Fugue is based on a single subject employed in some variation in each canon and fugue

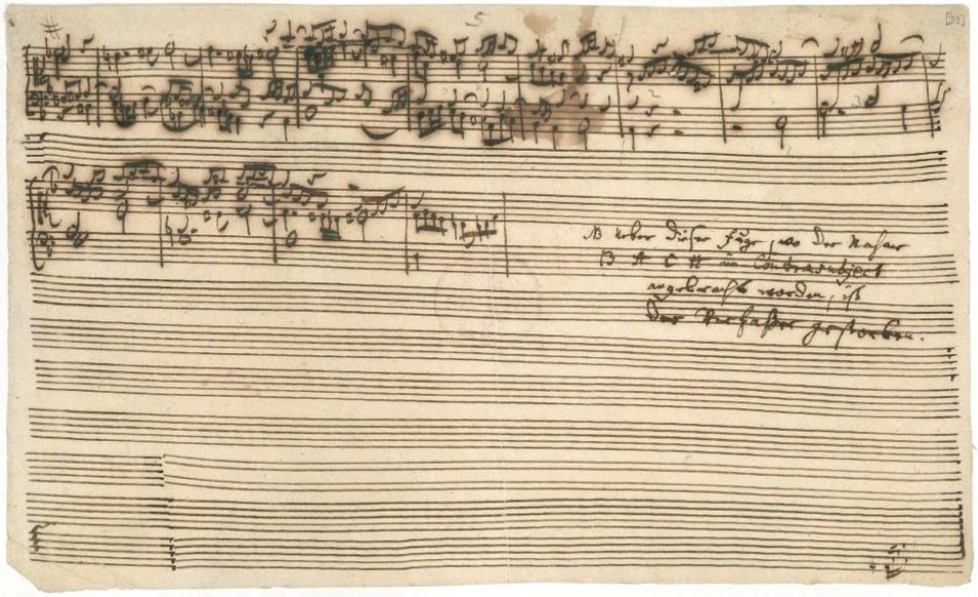
- **Simple fugues** (Contrapunctus I-IV, 4 voices)
- **Counter fugues** subject used simultaneously in regular, inverted, augmented, and diminished forms (Contrapunctus V- VII)
- **Double and triple fugue**s, employing two and three subjects respectively (Contrapunctus VIII – XI)
- **Mirror fugues**, a piece is notated once and then with voices and counterpoint completely inverted, without violating contrapuntal rules or musicality (Contrapunctus XII – XIII)
- **Canons**, labelled by interval and technique (Augmentationem in Contrario Motu, alla Ottava, Decima in Contrapunto alla Terza, Duodecima in Contrapunto alla Quinta)

http://www.washington.edu/news/2016/11/30/what-makes-bach-sound-like-bach-new-dataset-teaches-algorithms-classical-music/

. . . UNFINISHED FUGUE

Fuga a 3 Soggetti (Contrapunctus XIV):

- 4-voice triple fugue
- the third subject of which is based on the B A C H motif



« At the point where the composer introduces the name BACH in the countersubject to this fugue, the composer died. »

THE IMITATION GAME

Input data

Combinatorial problem

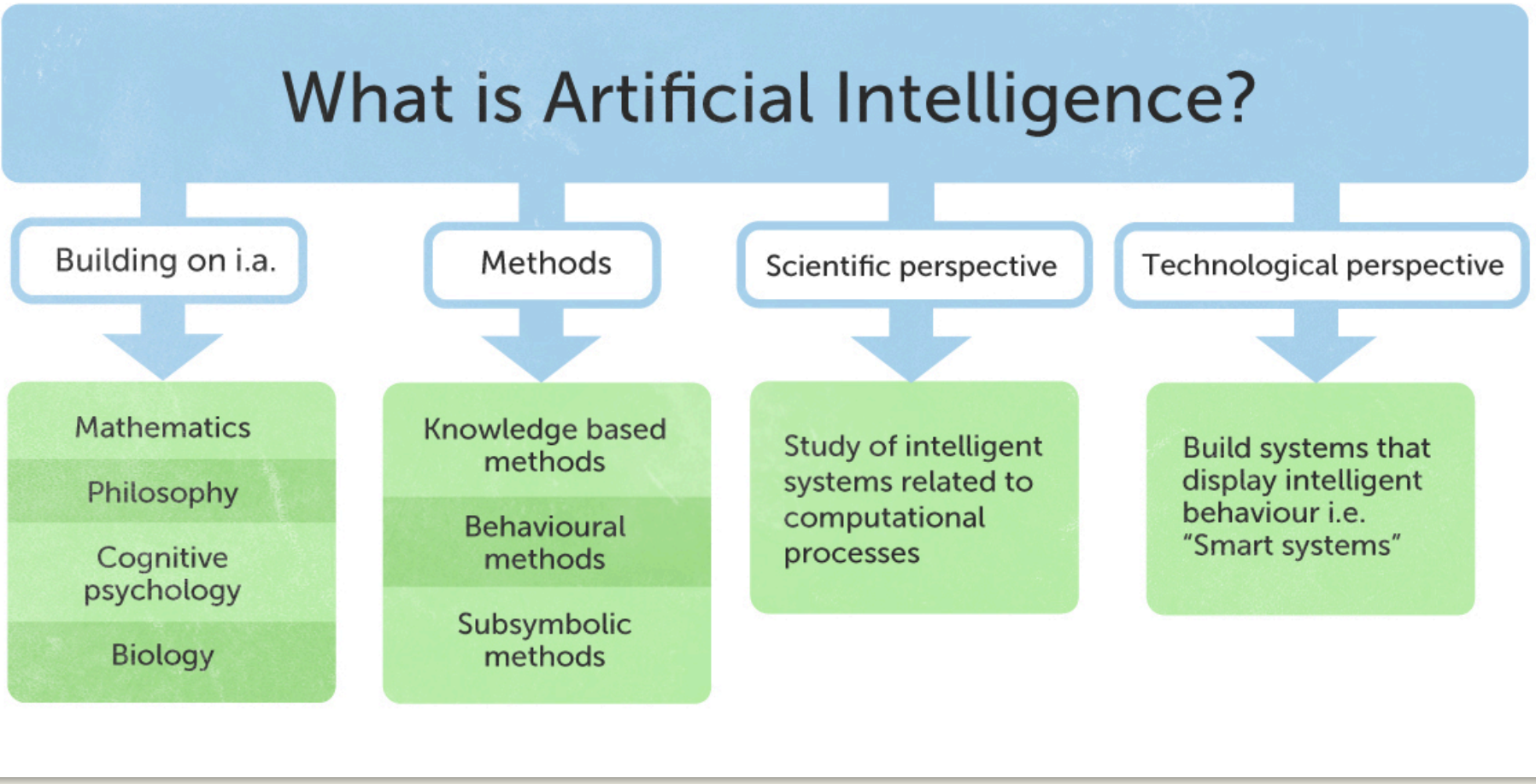
Critical variable

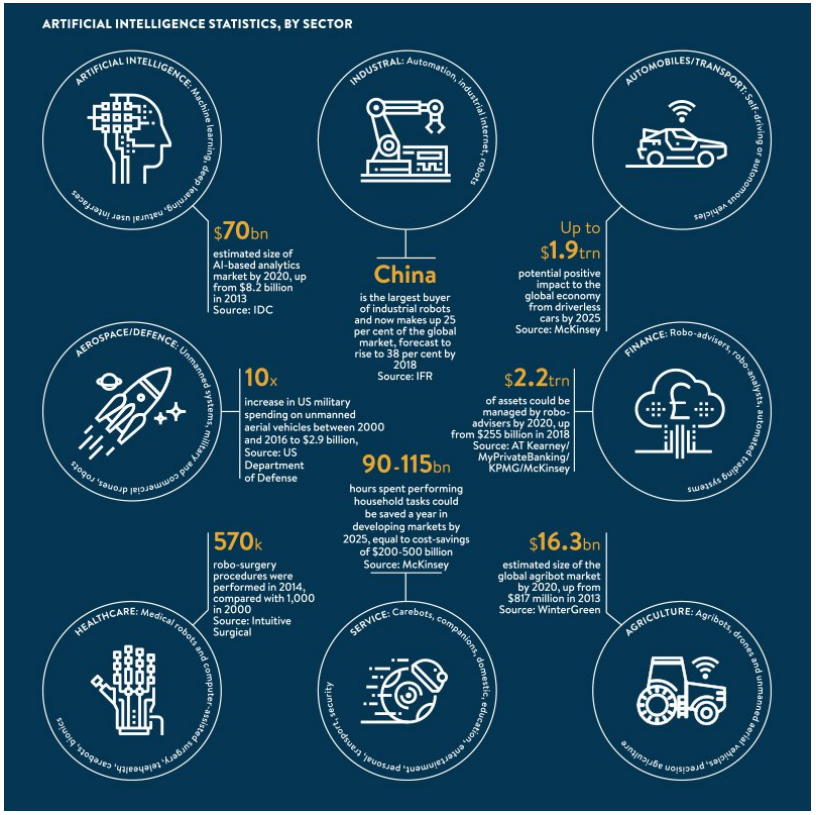
… CONTEMPORARY CHALLENGES

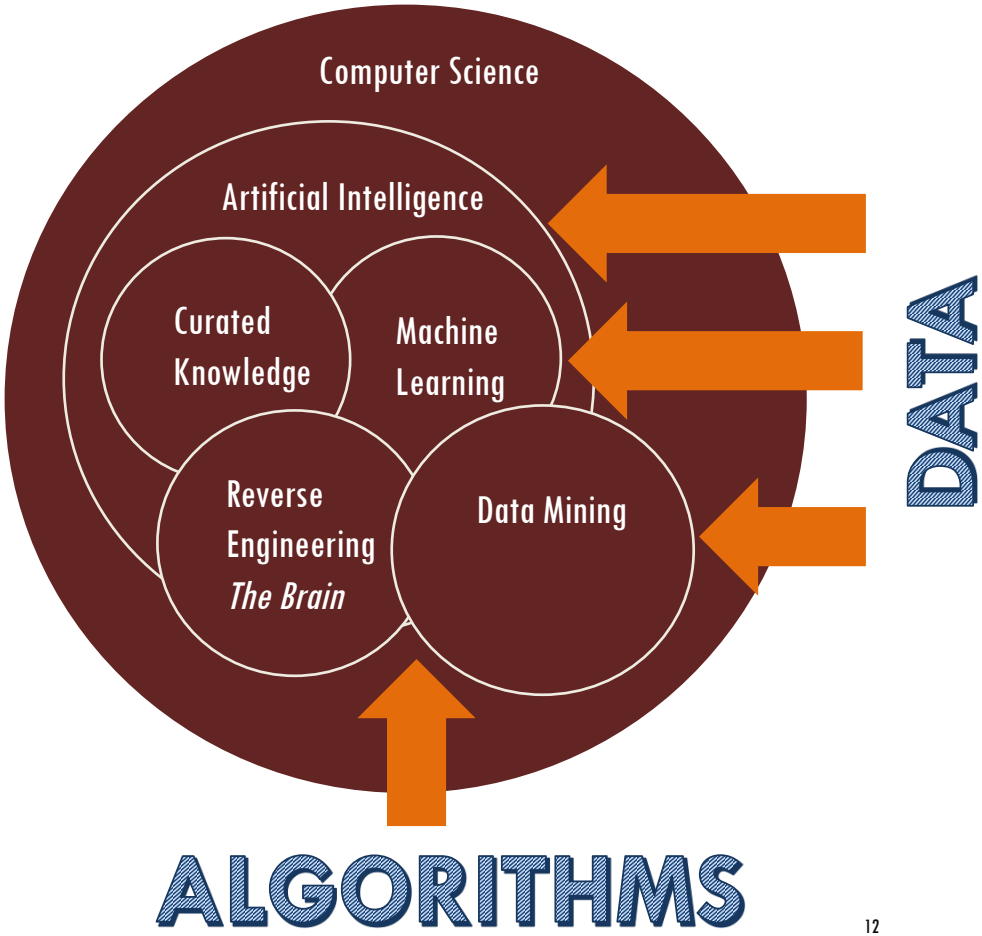
ARTIFICIAL INTELLIGENCE
DATA FOR INTELLIGENCE & INTELLIGENCE FOR DATA

https://ai100.stanford.edu/2016-report

What is Artificial Intelligence?



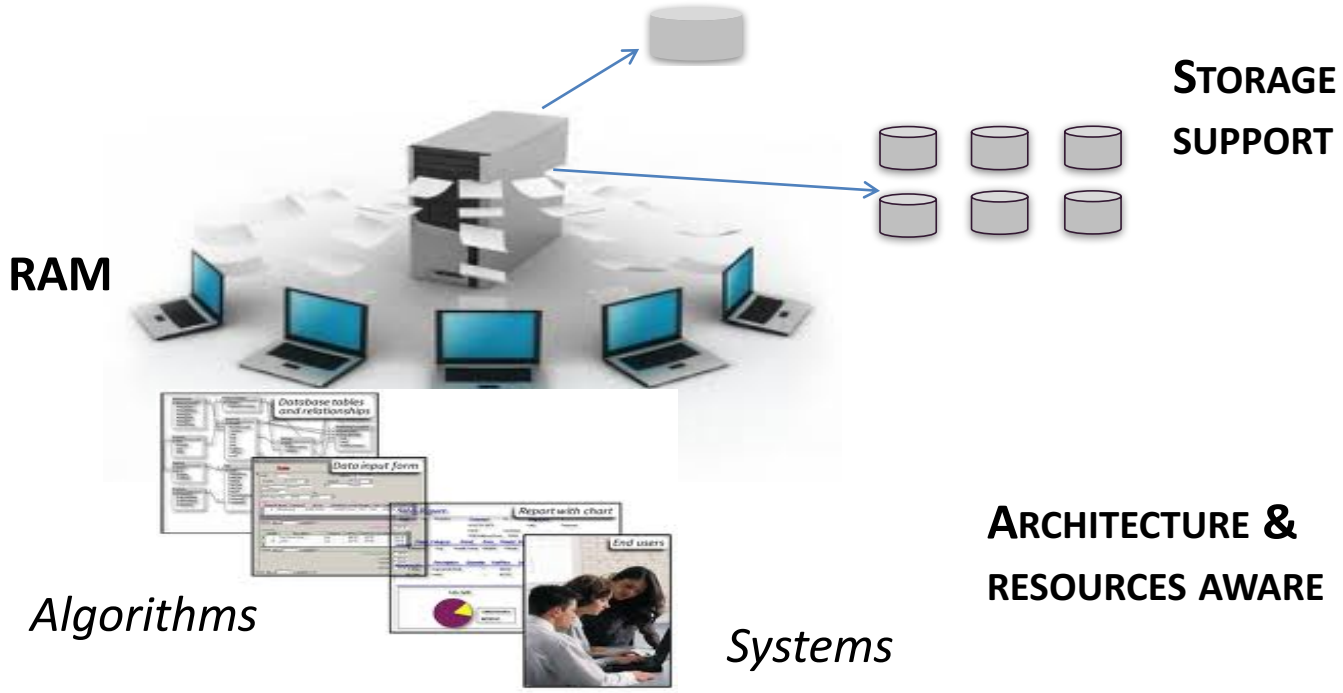

ARTIFICIAL INTELLIGENCE STATISTICS, BY SECTOR
ARTIFICIAL INTELLIGENCE: Machine learning, deep learning, natural user interfaces
$70bn
estimated size of AI-based analytics market by 2020, up from $8.2 billion in 2013
Source: IDC
INDUSTRAL: Automation, industrial internet, robots
China
is the largest buyer of industrial robots and now makes up 25 per cent of the global market, forecast to rise to 38 per cent by 2018
Source: IFR
AUTOMOBILES/TRANSPORT: Self-driving or autonomous vehicles
Up to $1.9trn
potential positive impact to the global economy from driverless cars by 2025
Source: McKinsey
AEROSPACE/DEFENCE: Unmanned systems, military and commercial drones, robots
10x
increase in US military spending on unmanned aerial vehicles between 2000 and 2016 to $2.9 billion,
Source: US Department of Defense
FINANCE: Robo-advisers, robo-analysts, algorithmic trading systems
$2.2trn
of assets could be managed by robo-advisers by 2020, up from $255 billion in 2018
Source: AT Kearney/ MyPrivateBanking/ KPMG/McKinsey
90-115bn
hours spent performing household tasks could be saved a year in developing markets by 2025, equal to cost-savings of $200-500 billion
Source: McKinsey
570k
robo-surgery procedures were performed in 2014, compared with 1,000 in 2000
Source: Intuitive Surgical
HEALTHCARE: Medical robots and computer-assisted surgery, telehealth, carebots, bionics
SERVICE: Carebots, companions, domestic, education, entertainment, personal, transport, security
$16.3bn
estimated size of the global agribot market by 2020, up from $817 million in 2013
Source: WinterGreen
AGRICULTURE: Agribots, drones and unmanned aerial vehicles, precision agriculture


Computer Science
Artificial Intelligence
Curated Knowledge
Machine Learning
Reverse Engineering
The Brain
Data Mining
DATA
ALGORITHMS

UNLIMITED COMPUTING RESOURCES & DATA

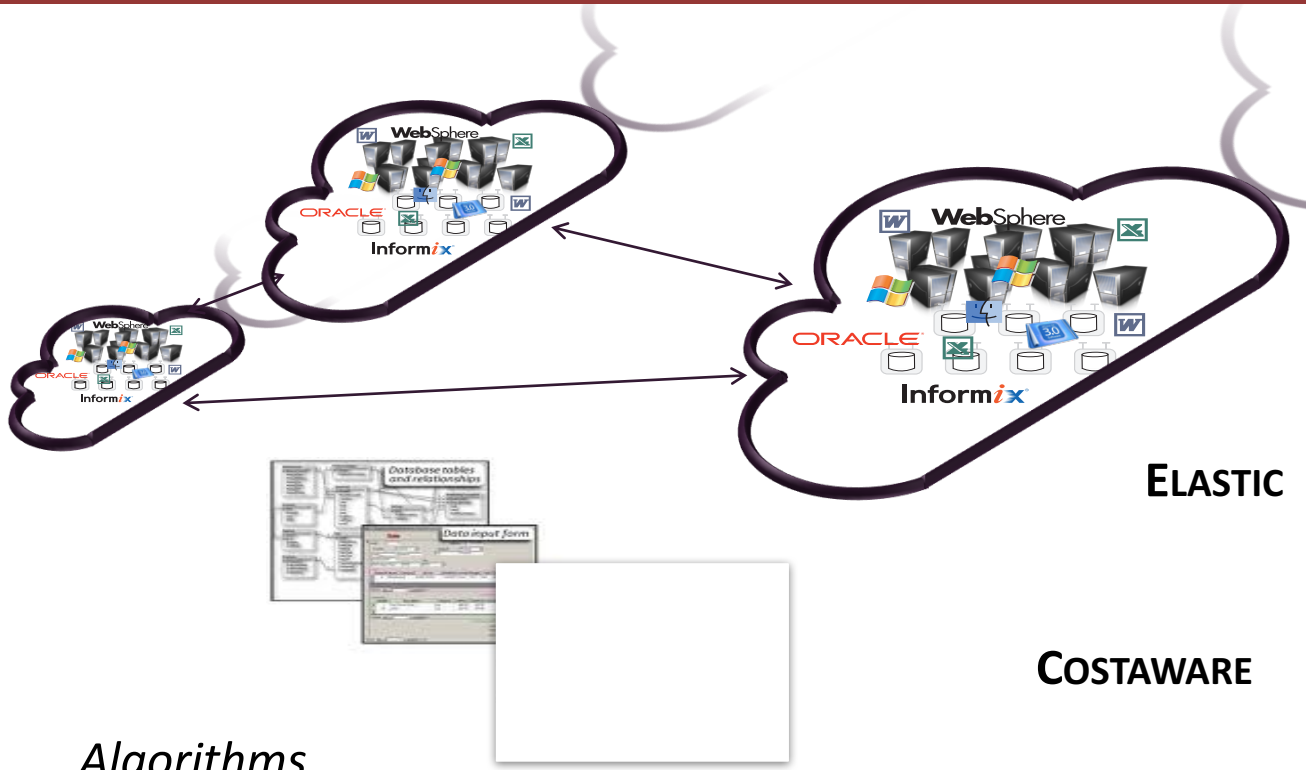
THE STORY OF DAVID & GOLIATH

DATA MANAGEMENT WITH RESOURCES CONSTRAINTS



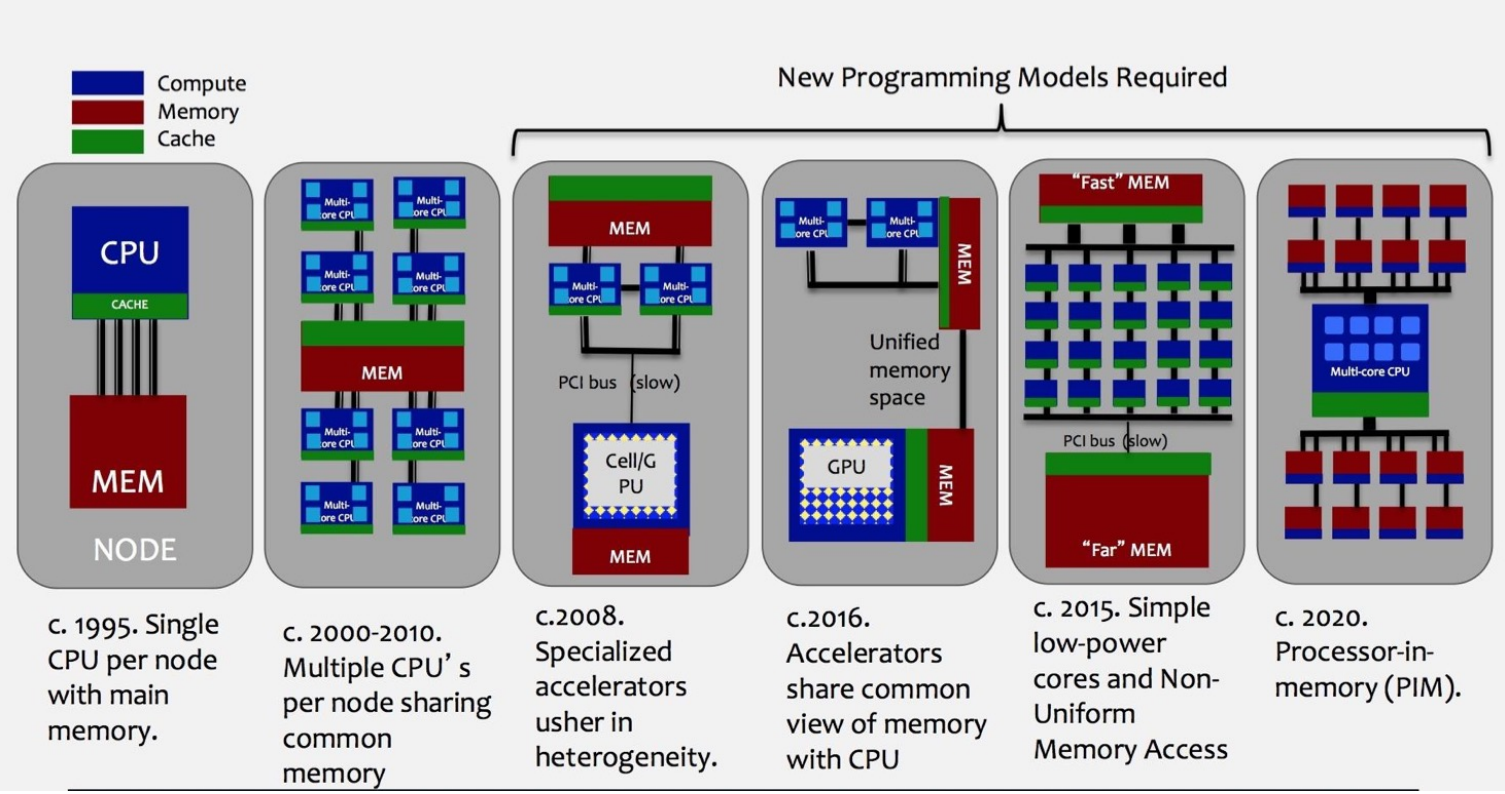
Efficiently manage and exploit data sets according to given specific storage, memory and computation resources

DATA MANAGEMENT WITHOUT RESOURCES CONSTRAINTS



Reduce the cost to manage and exploit data sets according to unlimited storage, memory and computation resources

EVOLUTION OF HPC NODE ARCHITECTURE



HPC PROCESSORS MARKET

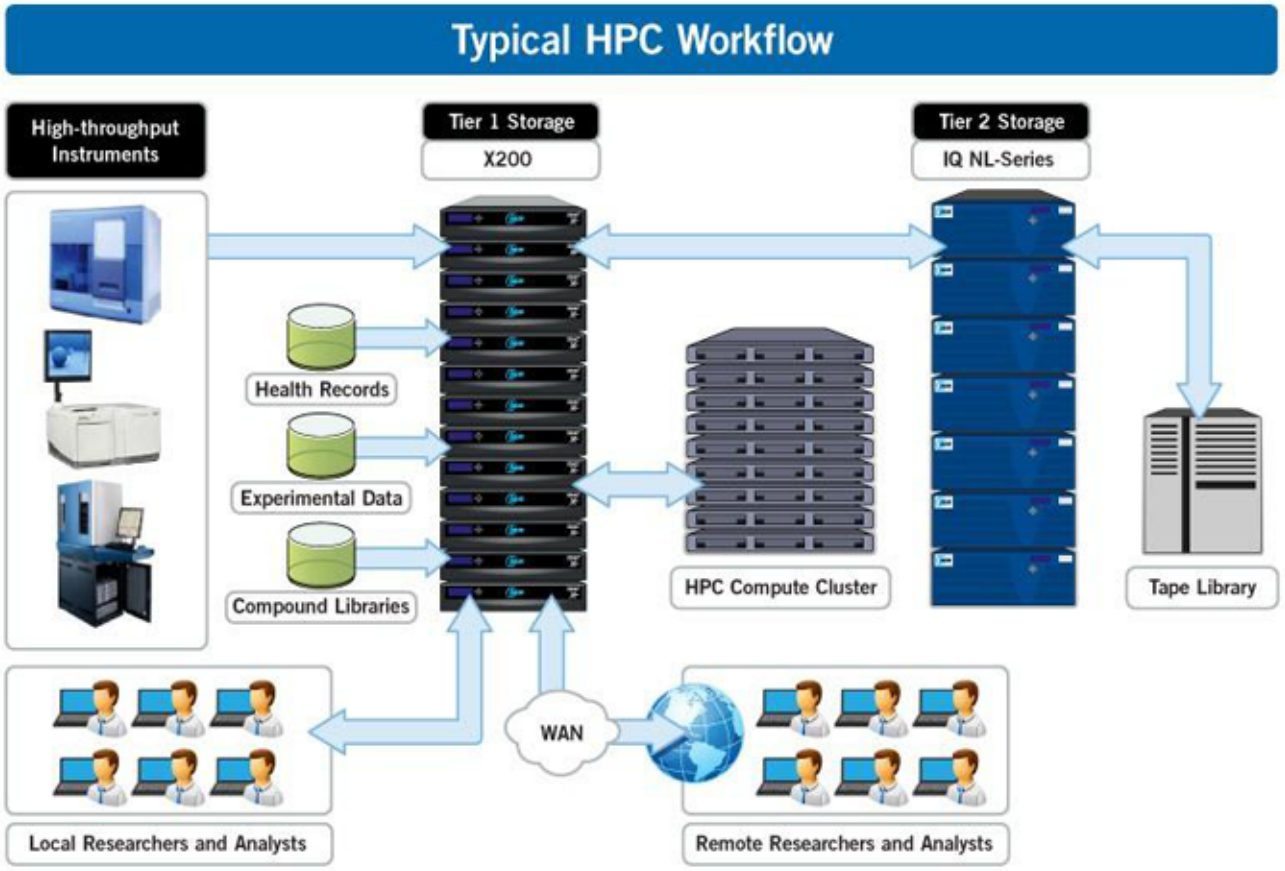
	2010	2014	2016	2017-2018
System Performance	20 Pflops	35 Pflops	100-200 Pflops	1EFlops
Power	6MW	10MW	10MW	20MW
Nodes	18,700	100,000	500,000	Million
Node Concurrency	12	32	Approx. 100	Approx. 1000
Interconnect Bandwidth	1.5GB/s	40GB/s	100 GB/says	200-400 GB/says
Mean Time To Interrupt (MTTI)	Days	Days	Days	Approx. a Day

HPC processors market Deep-dive, report published in 2016 by Blueocean Market Intelligence

	2010	2014	2016	2017-2018
System Performance	20 Pflops	35 Pflops	100-200 Pflops	1EFlops
Power	6MW	10MW	10MW	20MW
Node Concurrency	12	32	Approx. 100	Approx. 1000
Interconnect Bandwidth	1.5GB/s	40GB/s	100 GB/says	200-400 GB/says
Mean Time To Interrupt (MTTI)	Days	Days	Days	Approx. a Day

HOW TO BEST USE RESOURCES WHEN DEPLOYING ALOGORITHMS?

HPC processors market Deep-dive, report published in 2016 by Blueocean Market Intelligence


Typical HPC Workflow
High-throughput Instruments
Tier 1 Storage
X200
Tier 2 Storage
IQ NL-Series
Health Records
Experimental Data
Compound Libraries
HPC Compute Cluster
Tape Library
WAN
Local Researchers and Analysts
Remote Researchers and Analysts

THE CLOUD

Software as a service
PaaS

Platform as a service
PaaS

Infrastructure as a service
IaaS

- Illusion of infinite resources
- No up-front cost
- Fine-grained billing (e.g. hourly)

Promotes a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet

PaaS: allows customers to rent computers (virtual machines) on which to run their own computer applications.

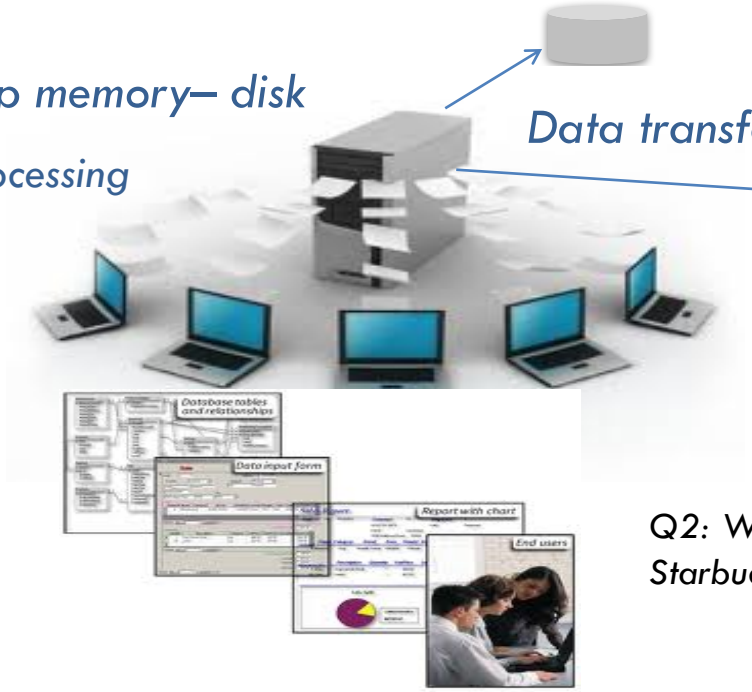
. . . WITH RESOURCES CONSTRAINTS

Distribution and organization of data on disk

Swap memory– disk

Data transfer

Query and data processing on server



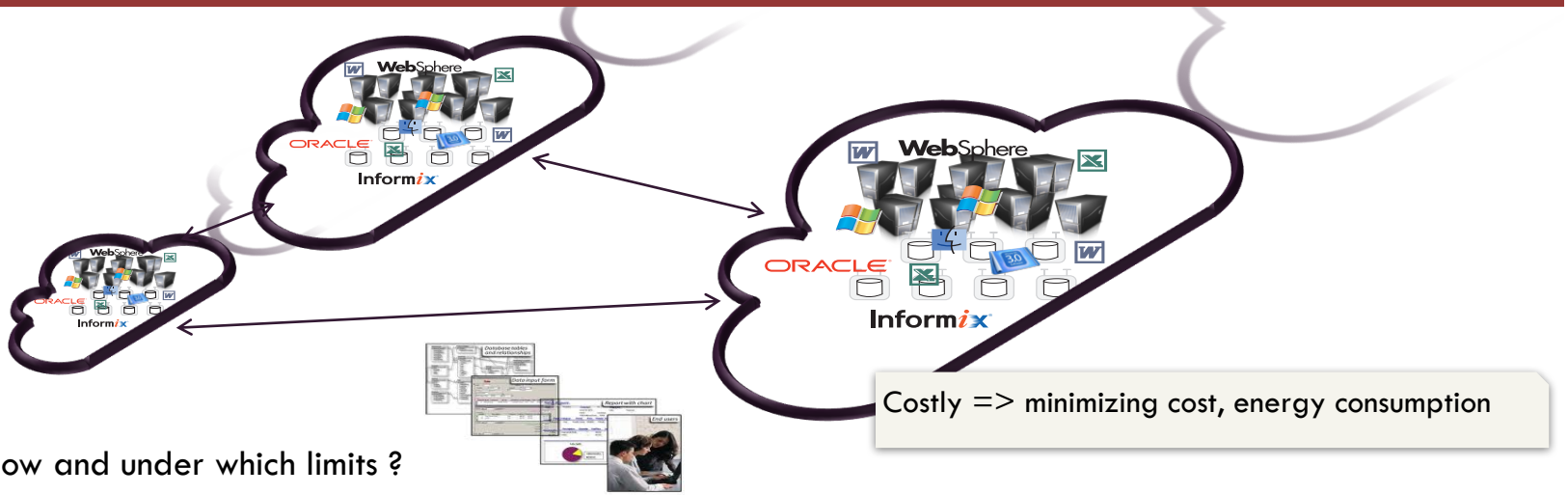
- Efficiency => time cost
- Optimizing memory and computing cost

Q2: Which are the consumption rules of Starbucks clients ?

Q1: Which are the most popular products at Starbucks ?

Efficiently manage and exploit data sets according to given specific storage, memory and computation resources

WITHOUT RESOURCES CONSTRAINTS . . .



- Query evaluation→ How and under which limits ?
- is not longer completely constraint by resources availability: computing, RAM, storage, network services
- Decision making process determined by resources consumption and consumer requirements

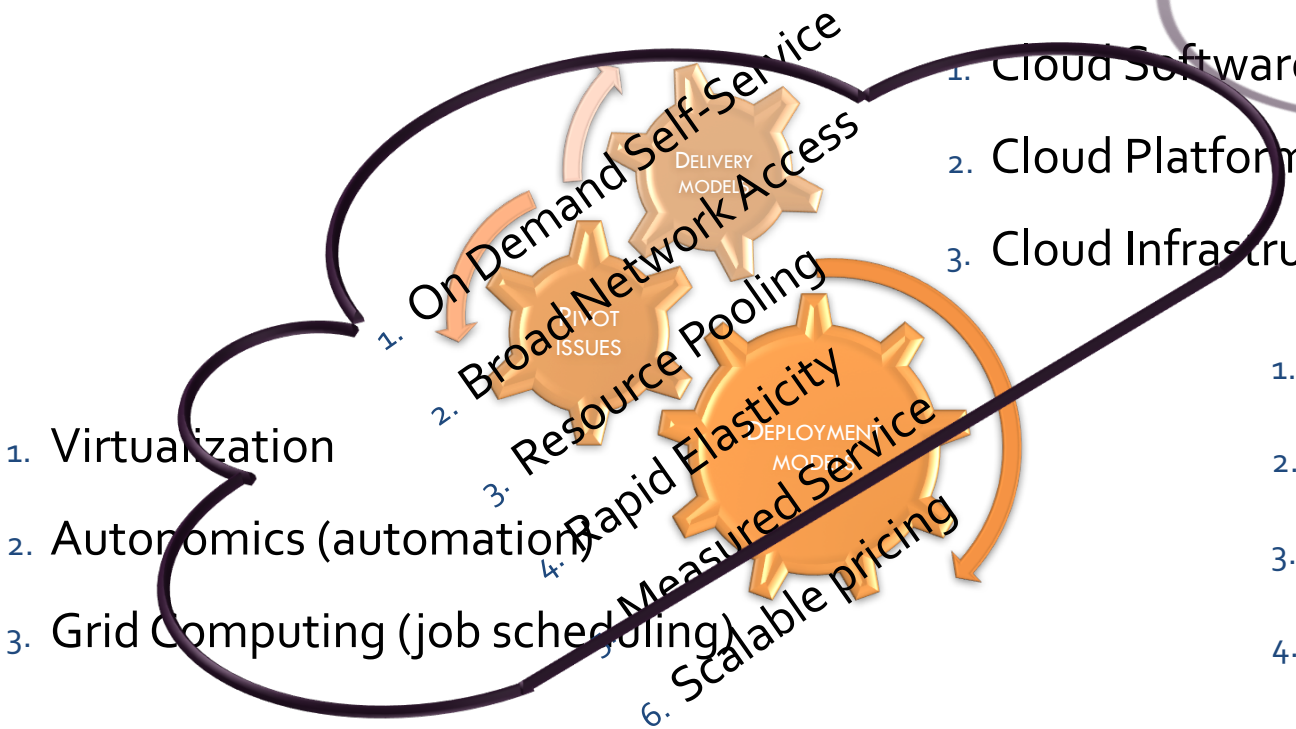
Data involved in the query, particularly in the result can have different costs: top 5 gratis and the rest available in return to a credit card number

Results storage and exploitation demands more resources

Reduce the cost to manage and exploit data sets according to unlimited storage, memory and computation resources

CLOUD IN A NUTSHELL

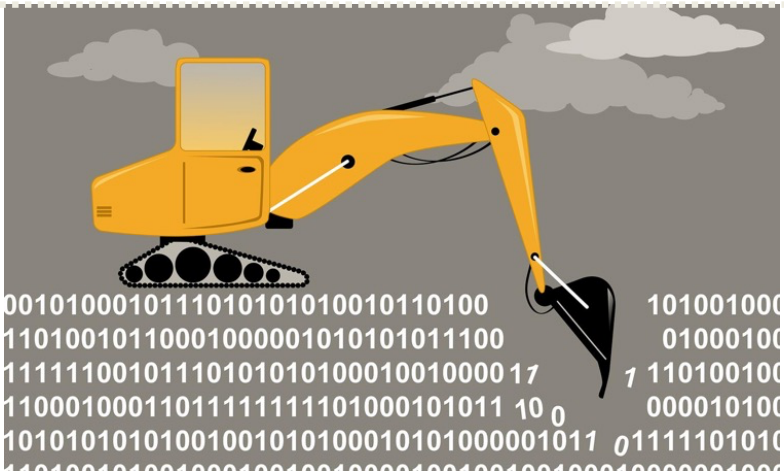
1. Virtualization
2. Autonomics (automation)
3. Grid Computing (job scheduling)



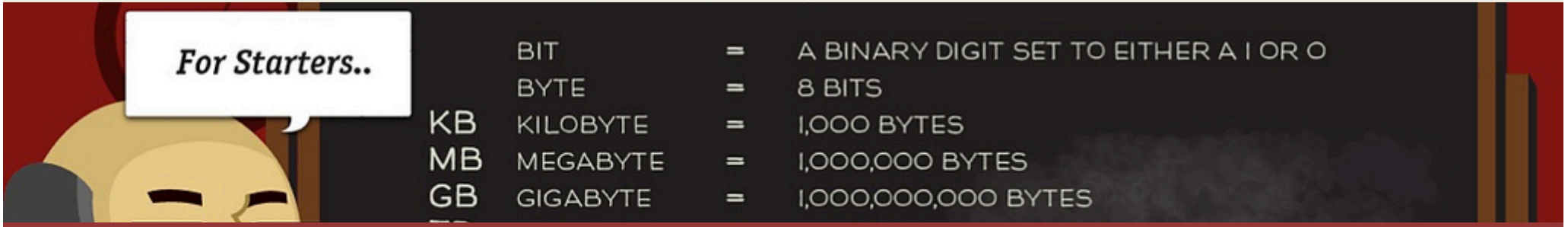
1. Cloud Software as a Service (SaaS)
2. Cloud Platform as a Service (PaaS)
3. Cloud Infrastructure as a Service (IaaS)

1. Private cloud
2. Public cloud
3. Hybrid cloud
4. Community Cloud

DATA SCIENCE: FROM BIG TO SMART

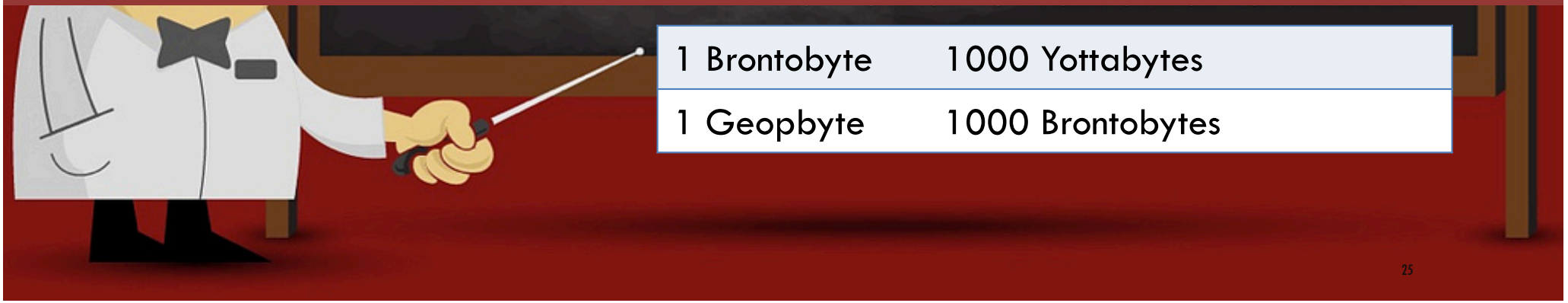


HOW BIG IS YOUR DATA ?

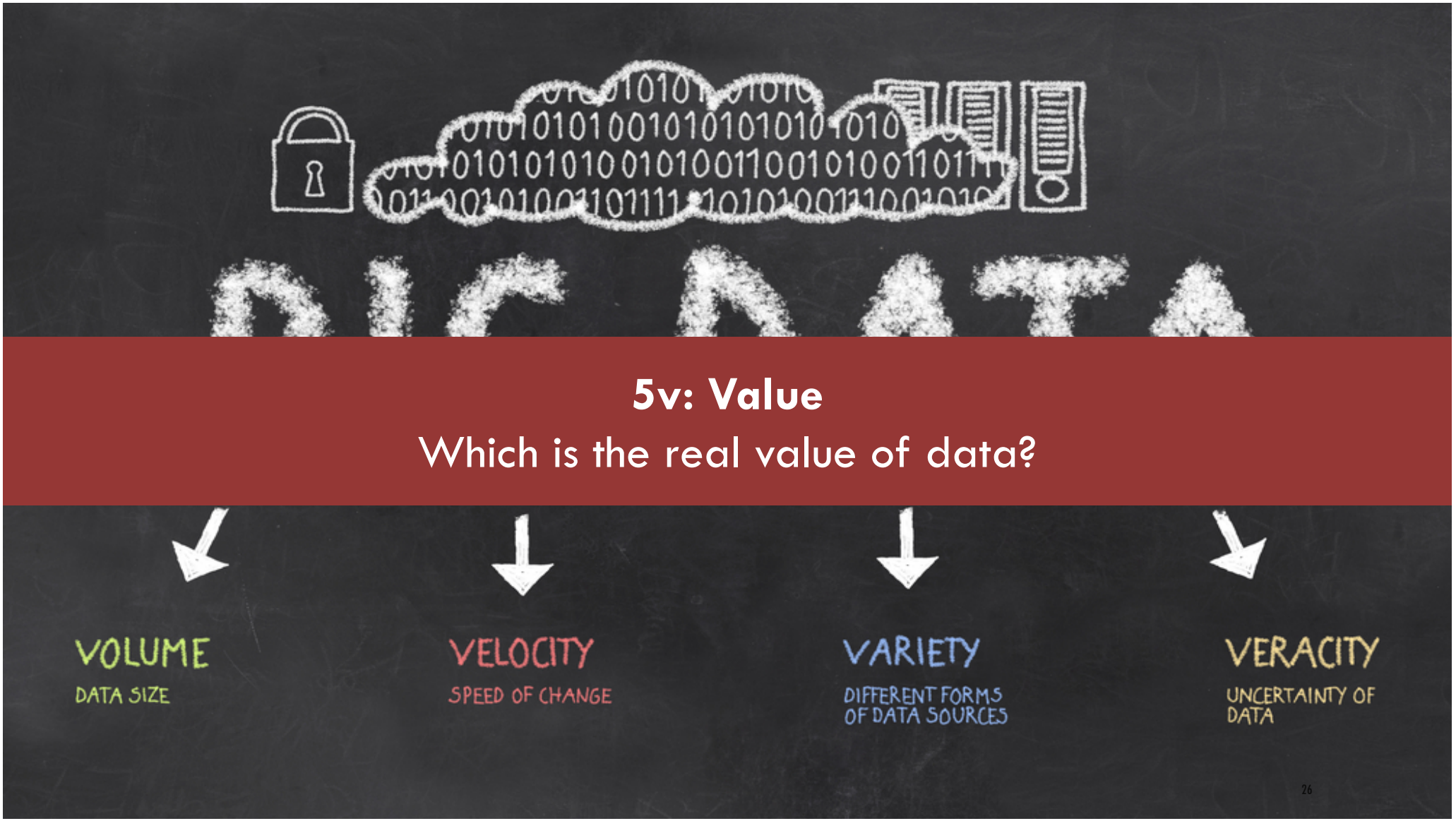


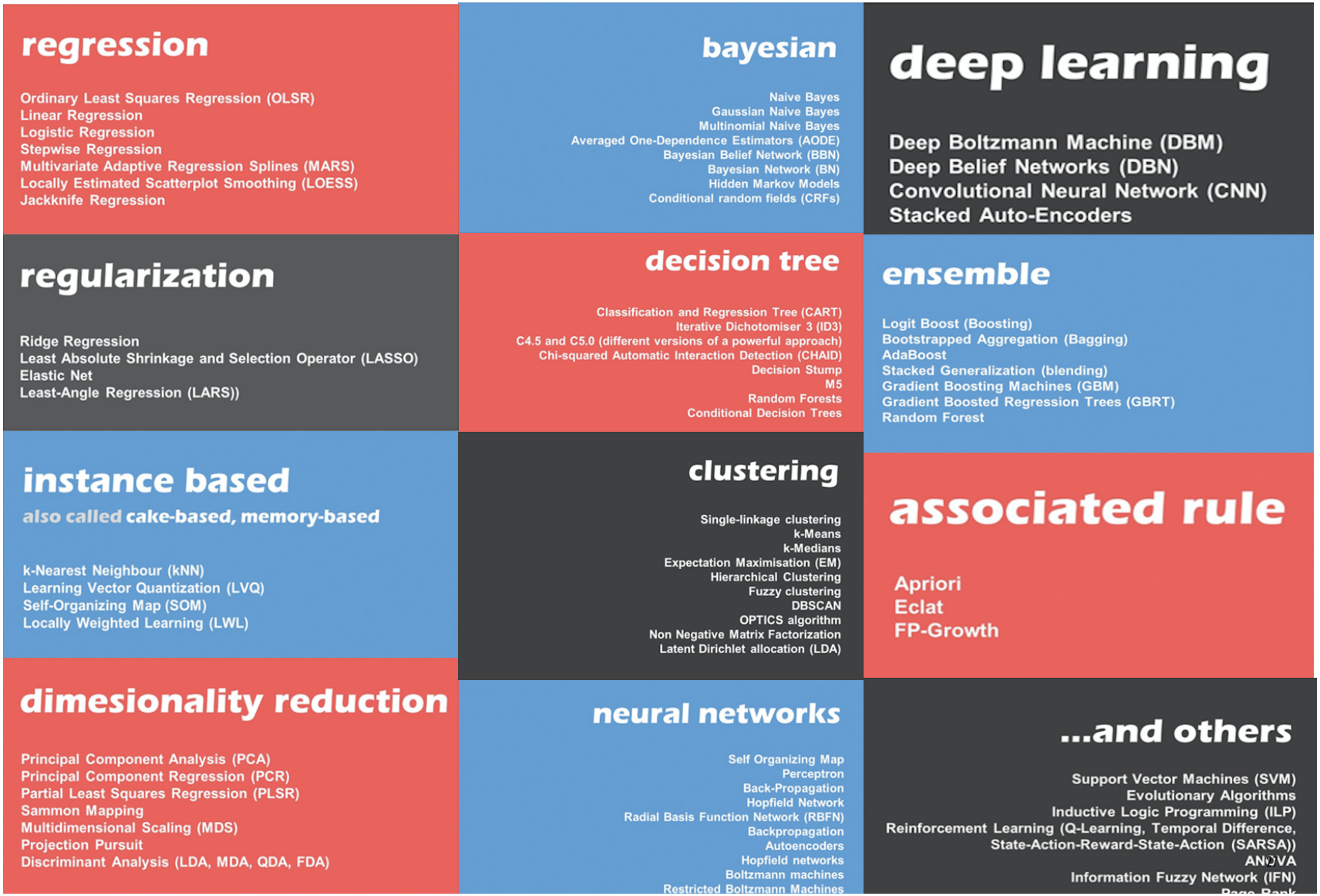
Helluva lot of data !!

http://spectrum.ieee.org/computing/software/beyond-just-big-data



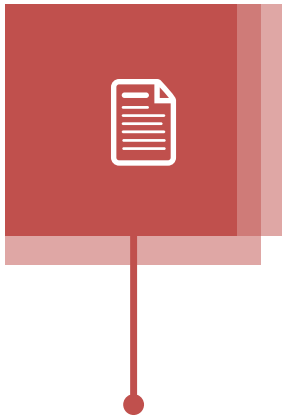
1 Brontobyte	1000 Yottabytes
1 Geopbyte	1000 Brontobytes


BIG DATA
5v: Value
Which is the real value of data?
VOLUME
DATA SIZE
VELOCITY
SPEED OF CHANGE
VARIETY
DIFFERENT FORMS OF DATA SOURCES
VERACITY
UNCERTAINTY OF DATA


regression
Ordinary Least Squares Regression (OLSR)
Linear Regression
Logistic Regression
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Jackknife Regression
bayesian
Naive Bayes
Gaussian Naive Bayes
Multinomial Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)
Bayesian Network (BN)
Hidden Markov Models
Conditional random fields (CRFs)
deep learning
Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders
regularization
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Elastic Net
Least-Angle Regression (LARS))
decision tree
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5 and C5.0 (different versions of a powerful approach)
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
M5
Random Forests
Conditional Decision Trees
ensemble
Logit Boost (Boosting)
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (blending)
Gradient Boosting Machines (GBM)
Gradient Boosted Regression Trees (GBRT)
Random Forest
instance based
also called cake-based, memory-based
k-Nearest Neighbour (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)
clustering
Single-linkage clustering
k-Means
k-Medians
Expectation Maximisation (EM)
Hierarchical Clustering
Fuzzy clustering
DBSCAN
OPTICS algorithm
Non Negative Matrix Factorization
Latent Dirichlet allocation (LDA)
associated rule
Apriori
Eclat
FP-Growth
dimesionality reduction
Principal Component Analysis (PCA)
Principal Component Regression (PCR)
Partial Least Squares Regression (PLSR)
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Discriminant Analysis (LDA, MDA, QDA, FDA)
neural networks
Self Organizing Map
Perceptron
Back-Propagation
Hopfield Network
Radial Basis Function Network (RBFN)
Backpropagation
Autoencoders
Hopfield networks
Boltzmann machines
Restricted Boltzmann Machines
...and others
Support Vector Machines (SVM)
Evolutionary Algorithms
Inductive Logic Programming (ILP)
Reinforcement Learning (Q-Learning, Temporal Difference,
State-Action-Reward-State-Action (SARSA))
ANOVA
Information Fuzzy Network (IFN)

Loosely Coupled

Data that withstands the breakdown into smaller pieces



Highly Structured

Hierarchical or other defined structures

Highly Related

Data extends out to many associations

Low Volatility

Relational Data tends to go through few and minor changes over time

Velocity

Volume

DATA

Variety

Value

Veracity

1000 Yottabytes	1 Brontobyte
1000 Brontobytes	1 Geopbyte

DATA CONSUMPTION PATTERNS

Document Transactional

Document Storage for cohesive and large transactional data

Document Archival

Document Storage for Archival Solutions

MapReduce

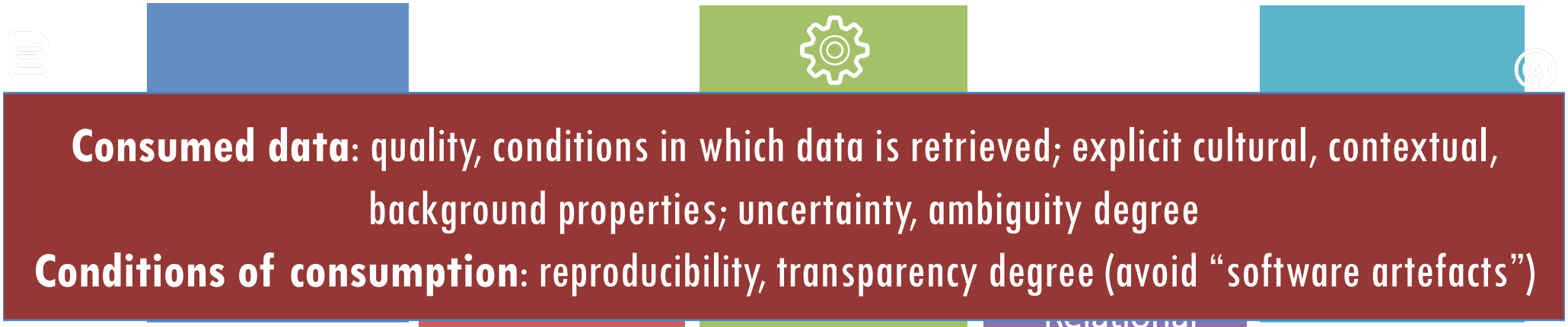
Data Processing for Complex BI and Reporting

Relational Transactional

Relational storage for highly structured transactional data

Streaming

Realtime processing and fulfillment

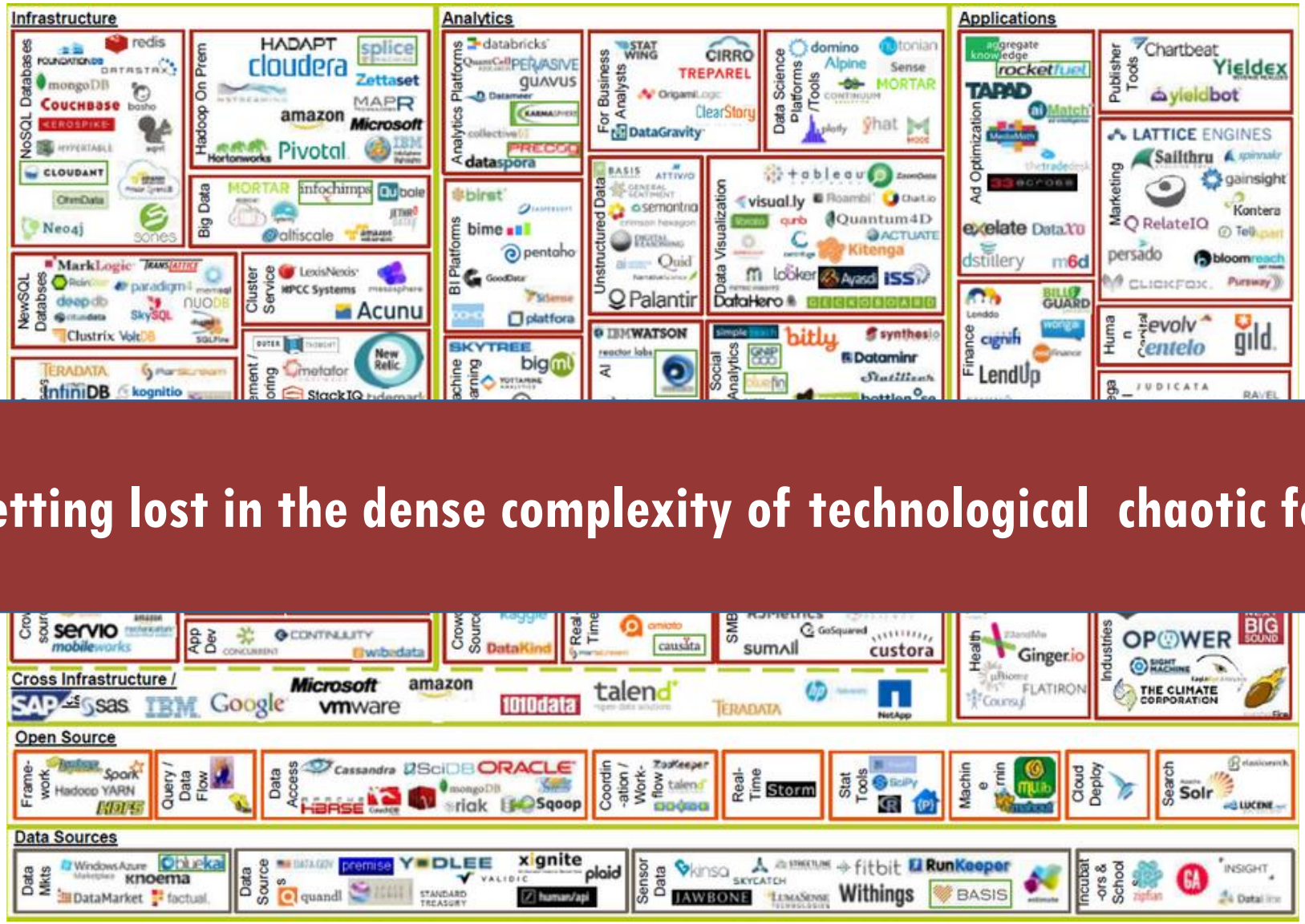


Document Archival

Document Storage for Archival Solutions

Transactional

Relational storage for highly structured transactional data


Infrastructure
Analytics
Applications
Cross Infrastructure /
Open Source
Data Sources
Avoid getting lost in the dense complexity of technological chaotic forest

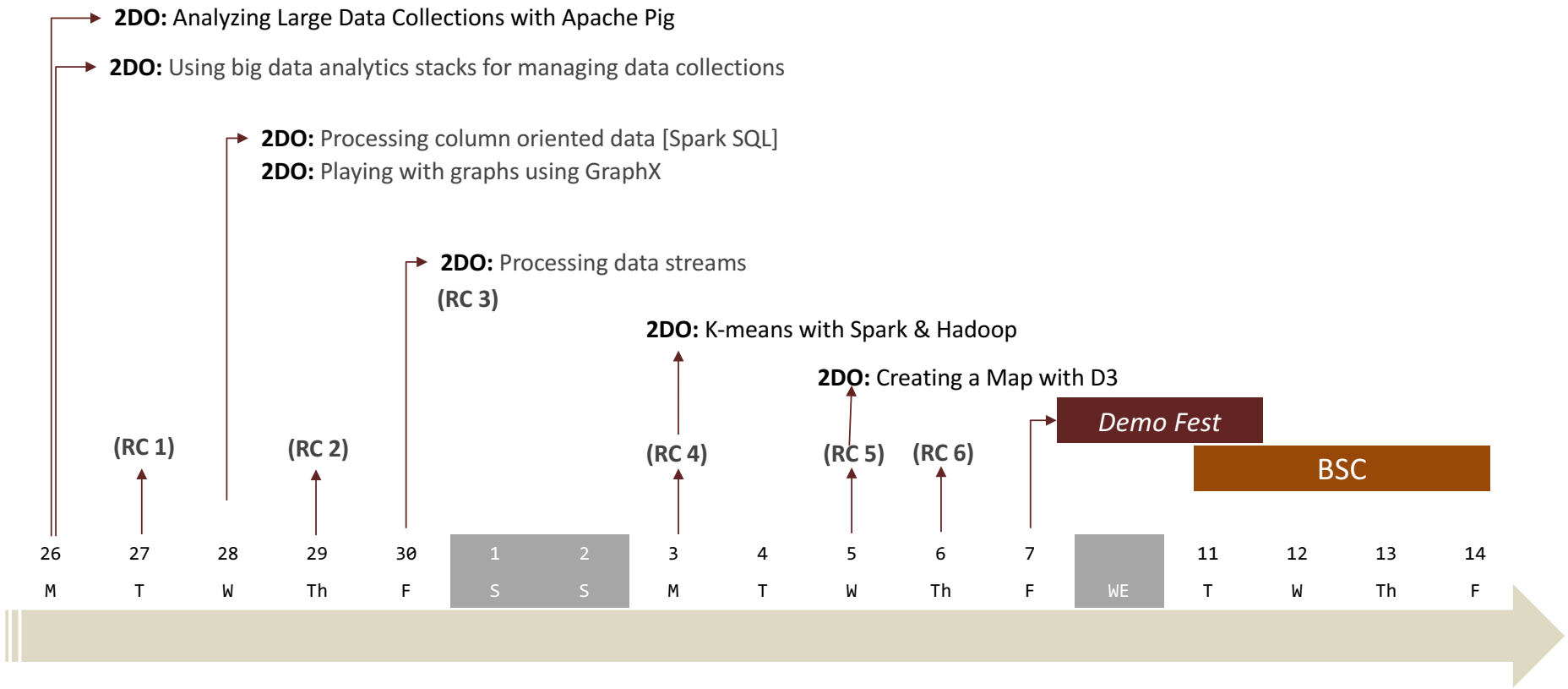
FINAL COMMENTS

Addressing **data centric sciences** problems is a matter of designing complex systems according to a **multidisciplinary vision**

Move from **design based on intuition & experience** to a more **formal and systematic way** to design systems

http://vargas-solar.com/datacentric-sciences/

SCHEDULE


2DO: Analyzing Large Data Collections with Apache Pig
2DO: Using big data analytics stacks for managing data collections
2DO: Processing column oriented data [Spark SQL]
2DO: Playing with graphs using GraphX
2DO: Processing data streams
(RC 3)
2DO: K-means with Spark & Hadoop
2DO: Creating a Map with D3
Demo Fest
BSC
(RC 1)
(RC 2)
(RC 4)
(RC 5)
(RC 6)
26 M
27 T
28 W
29 Th
30 F
1 S
2 S
3 M
4 T
5 W
6 Th
7 F
WE
11 T
12 W
13 Th
14 F

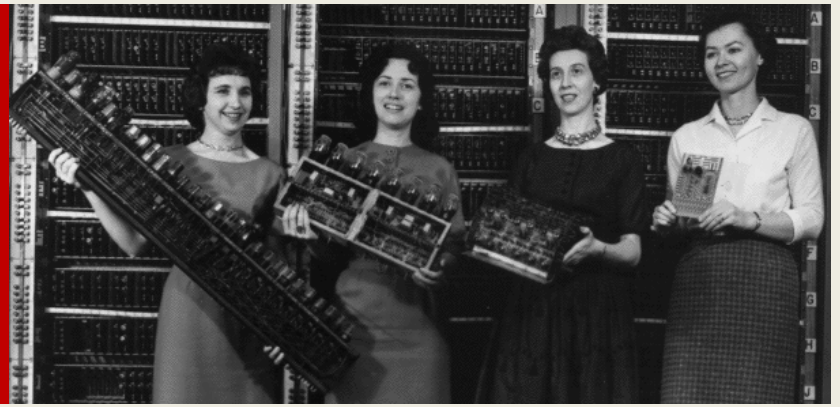
LEARNING METHODOLOGY & EVALUATION

- **Doing for learning**: theory with examples (lectures) and hands on (on computer)
- **Critical thinking & scientific debate**: reading, reading controls & collective discussion
- **Project:** problem statement (lecturers), design, solution approach & possibly implementation of some elements of the solution (teams)

Hands on: reports	50%
Reading control:	30%
Project:	20%


HOM
BRES
ROMPER CON ESTEREOTIPOS NOS TRANSFORMA
EN UNA SOCIEDAD + INCLUSIVA Y TOLERANTE

Women in Research



TRANSFORMING BIG INTO SMART

1 Data collection sizes

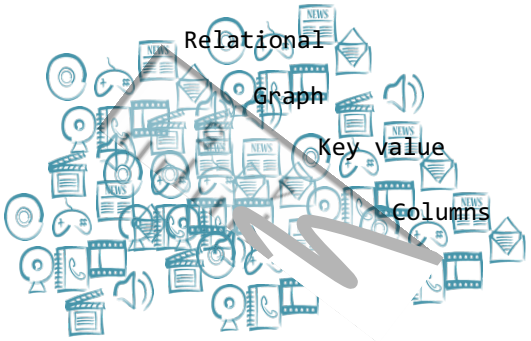
Yota 10^{24}

Zetta 10^{21}

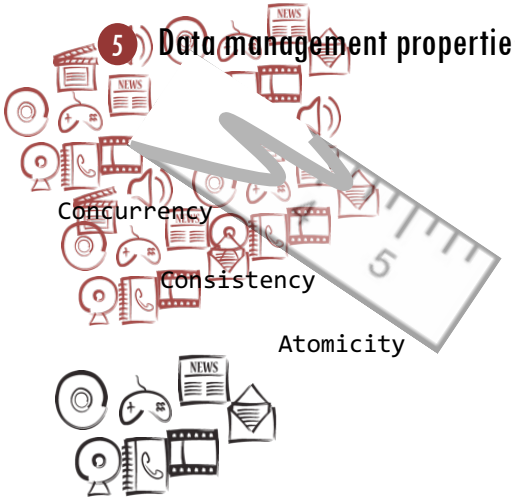
Exa 10^{18}

Peta 10^{15}

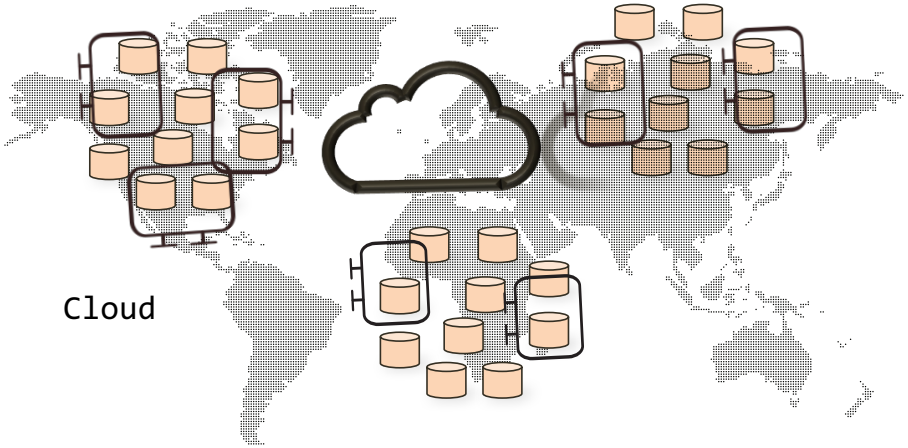
2 Data formats



5 Data management properties

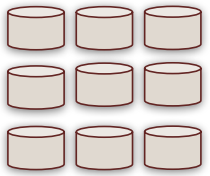


4 Data delivery mechanisms



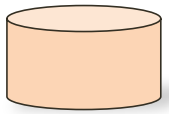
Cloud

RAID

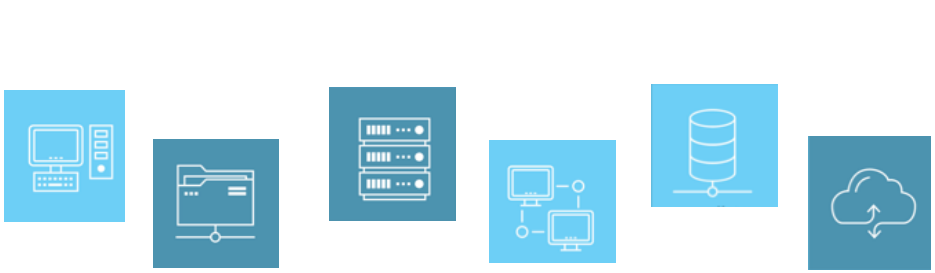


3 Data storage supports

Disk



Computing resources

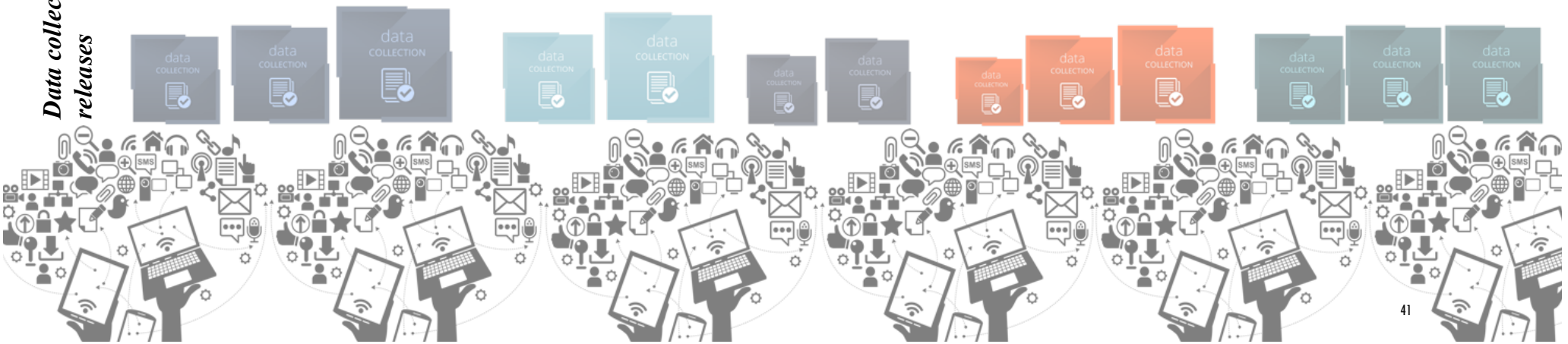


Applications & Data consumers

Data cleaning, processing and storage requires a lot of
DECISION MAKING

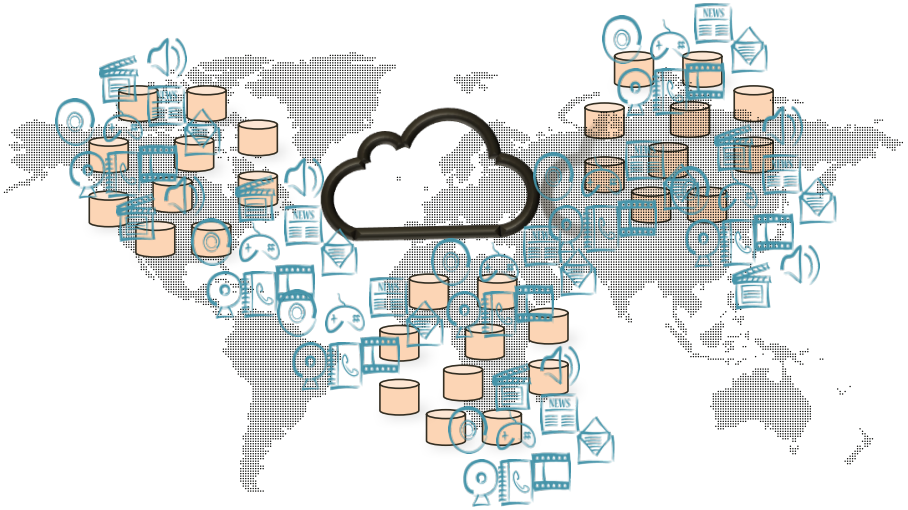
Data scientist requires **knowledge** about data collections content

Data collections' releases



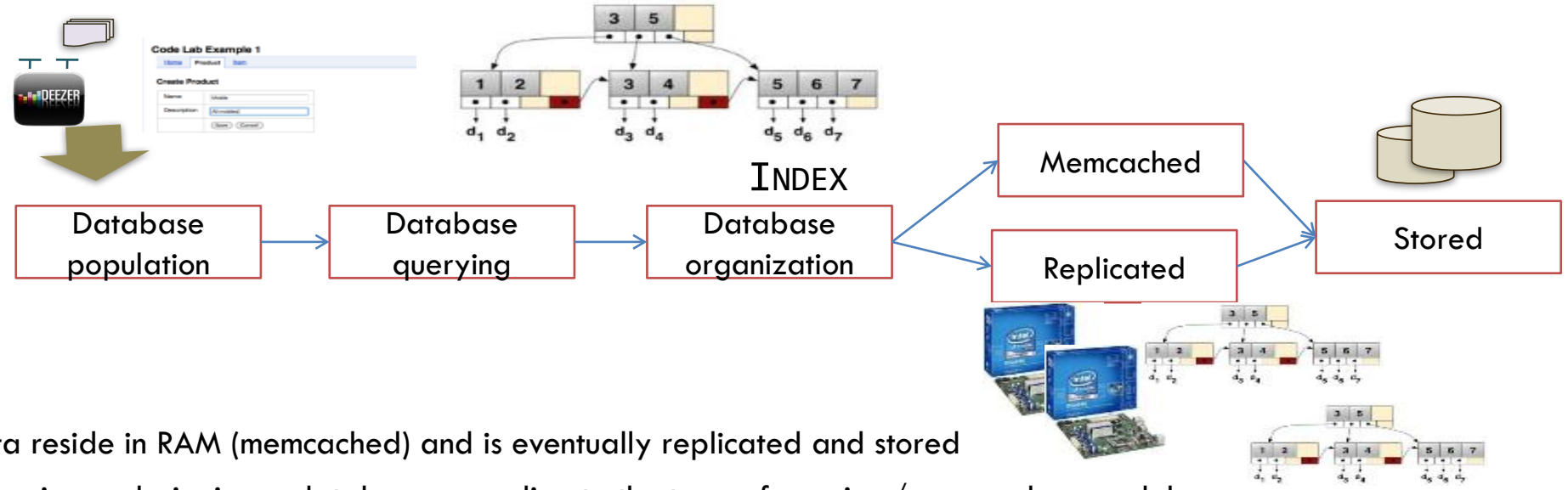
SCALING DATABASE SYSTEMS

- A system is scalable if increasing its resources (CPU, memory, disk) results in **increased performance** proportionally to the added resources
- Improving performance means serving more units of work for handling larger units of work like when data sets grow
- Database systems have been scaled by buying bigger faster and more expensive machines



- Vertically (SCALE UP)
  - Add resources (CPU, memory) to a single node in a system
- Horizontally (SCALE OUT)
  - Add more nodes to a system

NOSQL DESIGN AND CONSTRUCTION PROCESS



Data reside in RAM (memcached) and is eventually replicated and stored

Querying = designing a database according to the type of queries / map reduce model

"On demand" data management: the database is virtually organized per view (external schema) on cache and some view are made persistent

An elastic easy to evolve and explicitly configurable architecture

« MEMCHACHED »

«*memcached*» is a memory management protocol based on a cache:

- Uses the key-value notion
- Information is completly stored in RAM

«*memcached*» protocol for:

- Creating, retrieving, updating, and deleting information from the database
- Applications with their own «*memcached*» manager (Google, Facebook, YouTube, FarmVille, Twitter, Wikipedia)

PROBLEM STATEMENT: HOW MUCH TO GIVE UP?

Strict
The changes to the data are atomic and appear to take effect instantaneously. This is the highest form of consistency.

Sequential
Every client sees all changes in the same order they were applied.

Causal
All changes that are causally related are observed in the same order by all clients.

Eventual
When no updates occur for a period of time, eventually all updates will propagate through the system and all replicas will be consistent.

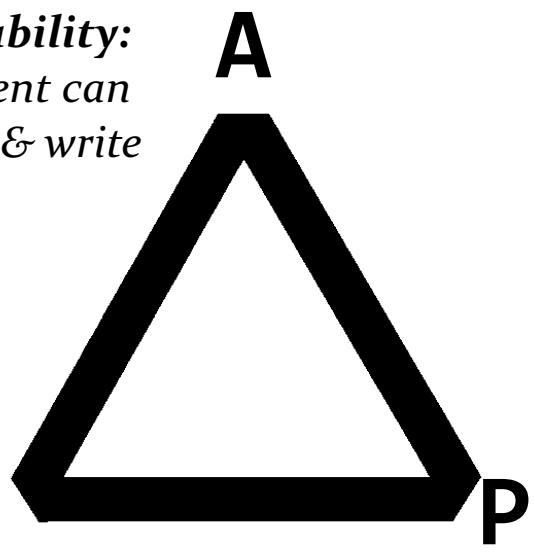
Weak
No guarantee is made that all updates will propagate and changes may appear out of order to various clients.

CAP theorem[1]: a system can have two of the three properties

NoSQL systems sacrifice **consistency**

1 Eric Brewer, "Towards robust distributed systems." PODC. 2000 http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

VISUAL GUIDE TO NOSQL SYSTEMS



Availability:
each client can always read & write

C - A

- RDBM's
- MySQL
- Postgres
- etc
- Aster Data
- GreenPlum
- Vertica

Consistency:
all clients always have the same view of de data

C - P

- BigTable
- HyperTable
- Hbase
- MongoDB
- TerraStore
- Scalaris

A - P

- Dynamo
- Voldemort
- Tokyo Cabinet
- KAI
- Cassandra
- SimpleDB
- CouchDB
- Riak

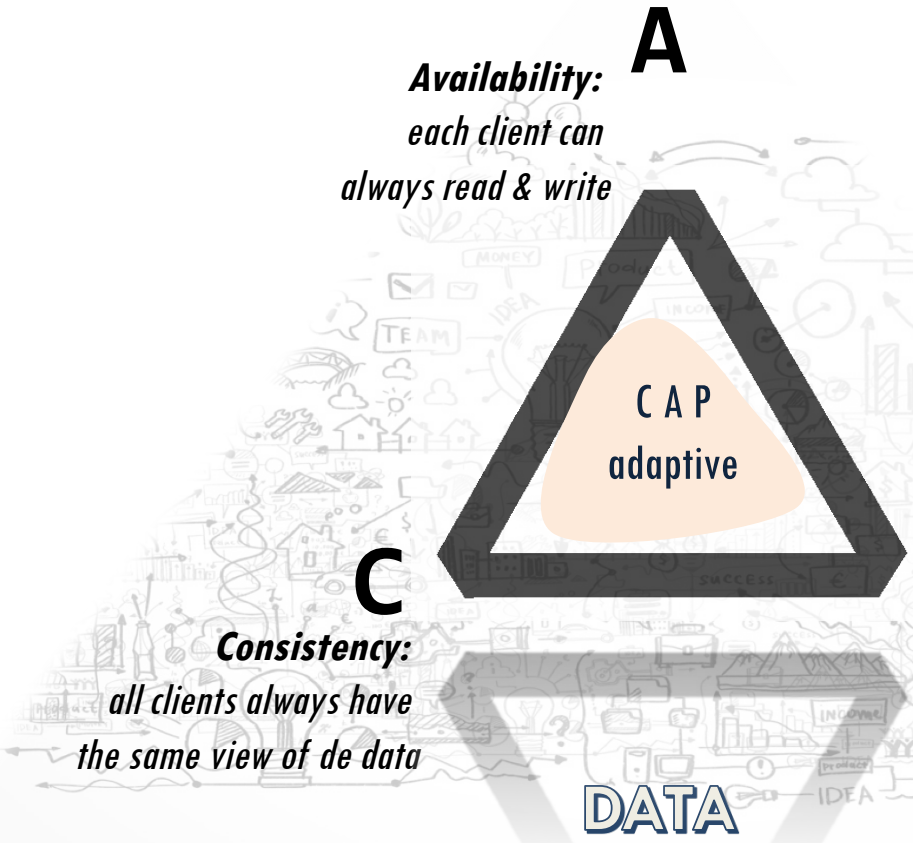
Partition tolerance:
The system works well despite physical network partitions

- BerkeleyDB
- MemcacheDB
- Redis

Data models

- Relational
- Key-Value
- Column oriented Tabular
- Document oriented

EFFICIENT R/W OF FRAGMENTED AND DISTRIBUTED DATA



Requirements

Operations

Hardware

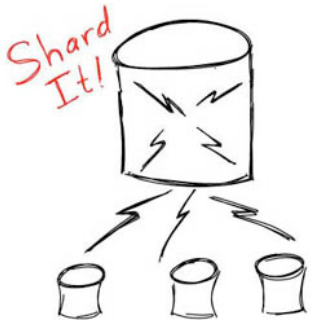
NOSQL STORES: AVAILABILITY & PERFORMANCE

Replication

- Copy data across multiple servers (each bit of data can be found in multiple servers)
- Increase data availability
- Faster query evaluation

Sharding

- Distribute different data across multiple servers
- Each server acts as the single source of a data subset



Orthogonal techniques

REPLICATION: PROS & CONS

Data is more available

- Failure of a site containing E does not result in unavailability of E if replicas exist

Performance

- Parallelism: queries processed in parallel on several nodes
- Reduce data transfer for local data

Increased updates cost

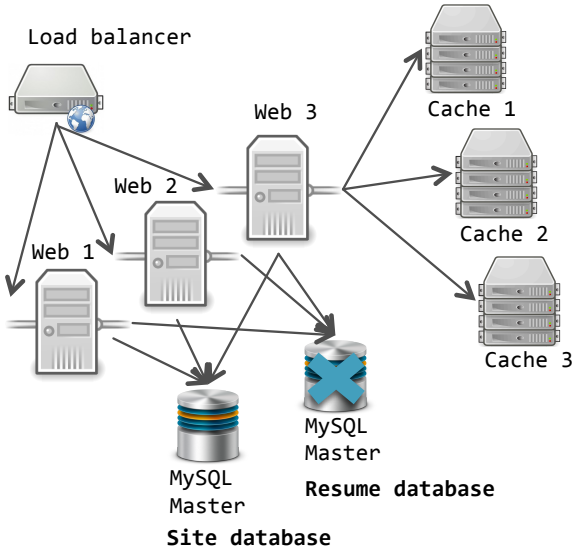
- Synchronisation: each replica must be updated

Increased complexity of concurrency control

- Concurrent updates to distinct replicas may lead to inconsistent data unless special concurrency control mechanisms are implemented

SHARDING: WHY IS IT USEFUL?

- Scaling applications by reducing data sets in any single databases
- Segregating data
- Sharing application data
- Securing sensitive data by isolating it



Improve read and write performance

- Smaller amount of data in each user group implies faster querying
- Isolating data into smaller shards accessed data is more likely to stay on cache
- More write bandwidth: writing can be done in parallel
- Smaller data sets are easier to backup, restore and manage

Massively work done

- Parallel work: scale out across more nodes
- Parallel backend: handling higher user loads
- Share nothing: very few bottlenecks

Decrease resilience improve availability

- If a box goes down others still operate
- But: Part of the data missing

SHARDING & REPLICATION

Sharding with no replication: unique copy, distributed data sets

- (+) Better concurrency levels (shards are accessed independently)
- (-) Cost of checking constraints, rebuilding aggregates
- Ensure that queries and updates are distributed across shards

Replication of shards

- (+) Query performance (availability)
- (-) Cost of updating, of checking constraints, complexity of concurrency control

Partial replication (most of the times)

- Only some shards are duplicated

SHARDING STRATEGIES USING MONGODB

MONGODB SHARDING

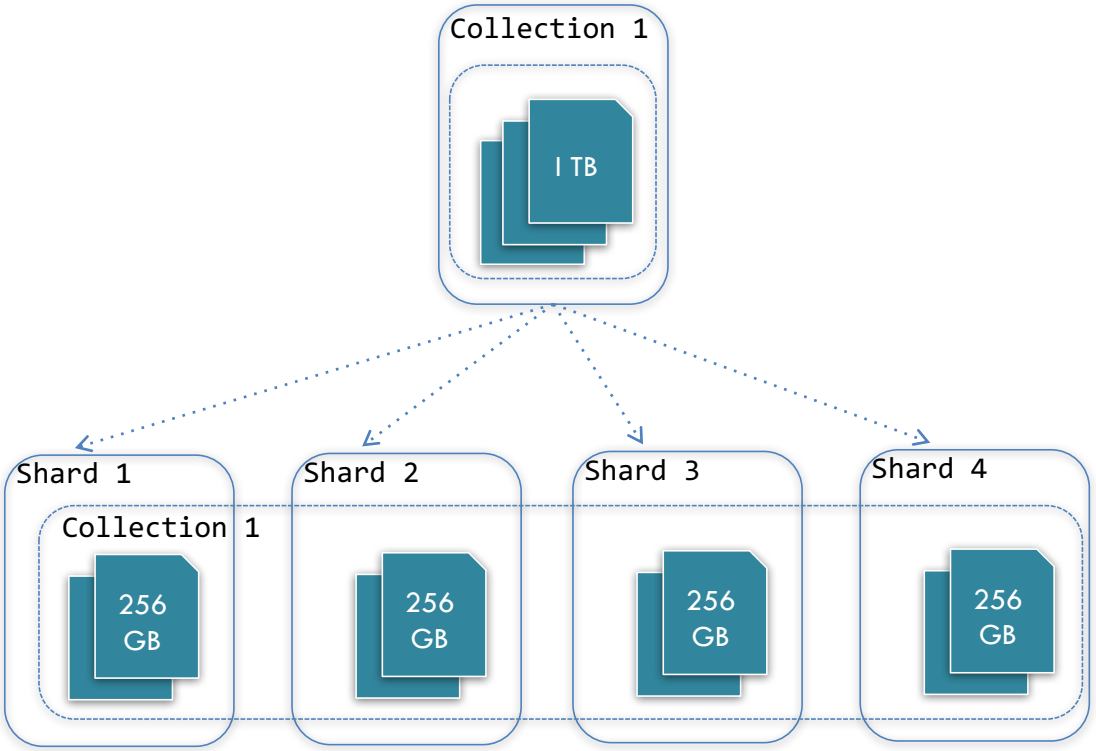
Database systems with large data sets and high throughput applications can challenge the capacity of a single server.

- High query rates can exhaust the CPU capacity of the server.
- Larger data sets exceed the storage capacity of a single machine.
- Finally, working set sizes larger than the system's RAM stress the I/O capacity of disk drives.

To address these issues of scales, database systems have two basic approaches: vertical scaling and sharding

- Vertical scaling adds more CPU and storage resources to increase capacity.
  - Scaling by adding capacity has limitations: high performance systems with large numbers of CPUs and large amount of RAM are disproportionately more expensive than smaller systems.
  - Additionally, cloud-based providers may only allow users to provision smaller instances.
  - As a result there is a practical maximum capability for vertical scaling.
- Sharding, or horizontal scaling, divides the data set and distributes the data over multiple servers, or shards.
  - Each shard is an independent database,
  - Collectively, the shards make up a single logical database

SHARDING DATA



SHARDING

Addresses the challenge of scaling to support high throughput and large data sets

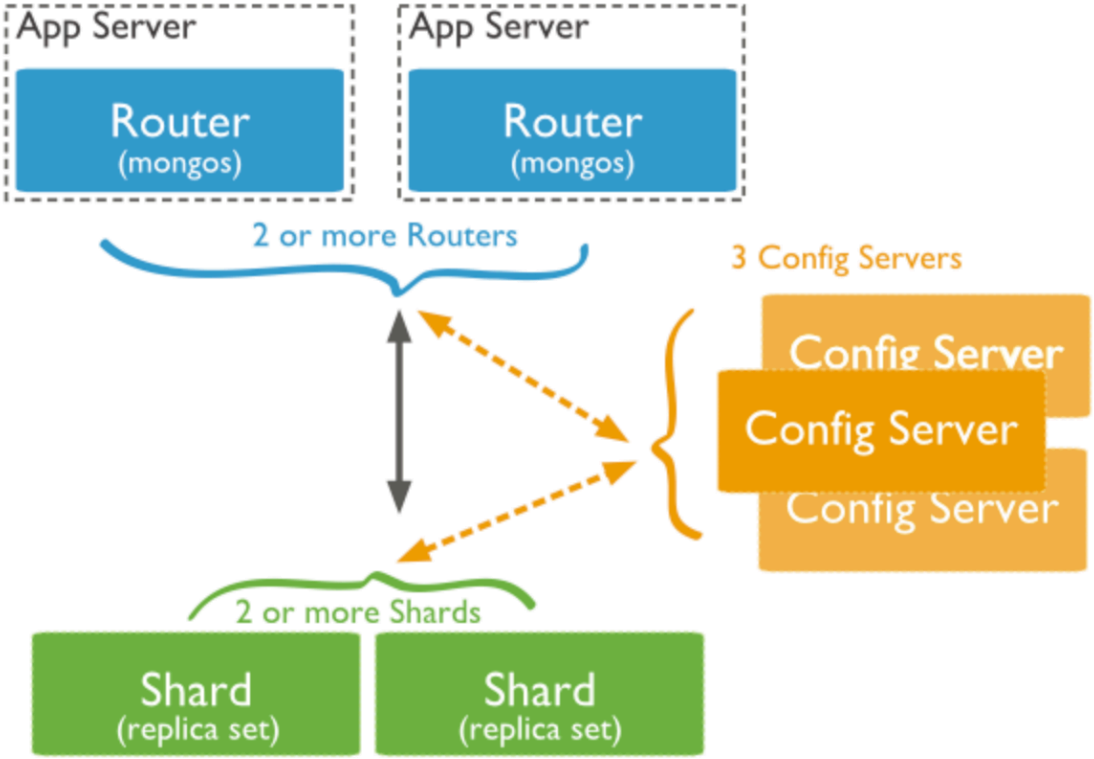
Reduces the number of operations each shard handles

- Each shard processes fewer operations as the cluster grows.
- As a result, a cluster can increase capacity and throughput horizontally
- For example, to insert data, the application only needs to access the shard responsible for that record

Reduces the amount of data that each server needs to store

- Each shard stores less data as the cluster grows.
- For example, if a database has a 1 terabyte data set, and there are 4 shards, then each shard might hold only 256GB of data. If there are 40 shards, then each shard might hold only 25GB of data.

SHARDING IN MONGODB



SHARDING IN MONGODB

Shards (mongd) store the data. To provide high availability and data consistency, in a production sharded cluster, each shard is a replica set.

Query Routers (**mongos** instances), interface with client applications and direct operations to the appropriate shard or shards.

- The query router processes and targets operations to shards and then returns results to the clients.
- A sharded cluster can contain more than one query router to divide the client request load.
- A client sends requests to one query router. Most sharded clusters have many query routers.

Config servers store the cluster's metadata.

- This data contains a mapping of the cluster's data set to the shards.
- The query router uses this metadata to target operations to specific shards.
- Production sharded clusters have exactly 3 config servers

PRIMARY SHARD

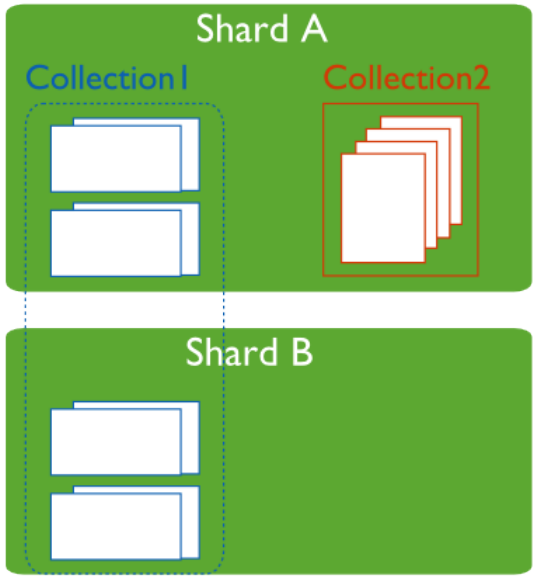
Every database has a “primary” shard that holds all the un-sharded collections in that database

To change the primary shard for a database, use the `movePrimary` command.

- The process of migrating the primary shard may take significant time to complete, and you should not access the collections until it completes.

When a new sharded cluster is deployed with shards that were previously used as replica sets, all existing databases continue to reside on their original shard

Databases created subsequently may reside on any shard in the cluster



CONFIG SERVERS

Special mongod instances that store the metadata for a sharded cluster.

Use a two-phase commit to ensure immediate consistency and reliability.

Do not run as replica sets.

All config servers must be available to deploy a sharded cluster or to make any changes to cluster metadata.

A production sharded cluster has exactly three config servers

- For testing purposes you may deploy a cluster with a single config server
- But to ensure redundancy and safety in production, you should always use three.

CONFIG DATABASE

Config servers store the metadata in the config database. The mongos instances cache this data and use it to route reads and writes to shards

MongoDB only writes data to the config server in the following cases:

- To create splits in existing chunks.
- To migrate a chunk between shards.

MongoDB reads data from the config server data in the following cases:

- A new mongos starts for the first time, or an existing mongos restarts.
- After a chunk migration, the mongos instances update themselves with the new cluster metadata.
- MongoDB also uses the config server to manage distributed locks.

DATA PARTITIONING

MongoDB distributes data, or shards, at the collection level.

Sharding partitions a collection's data by the shard key.

A shard key is either an indexed field or an indexed compound field that exists in every document in the collection.

MongoDB divides the shard key values into chunks and distributes the chunks evenly across the shards.

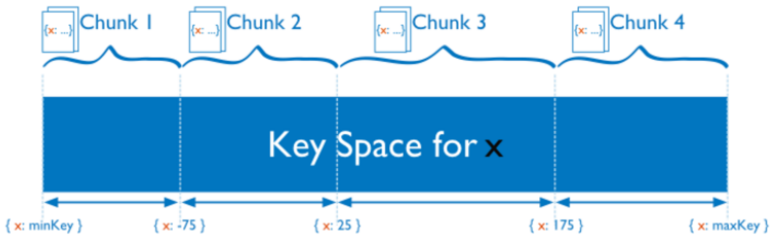
- range based partitioning or hash based partitioning

RANGE BASED SHARDNIG

MongoDB divides the data set into ranges determined by the shard key values

- Consider a numeric shard key: If you visualize a number line that goes from negative infinity to positive infinity, each value of the shard key falls at some point on that line.
- MongoDB partitions this line into smaller, non-overlapping ranges called chunks
- chunk is range of values from some minimum value to some maximum value.

Given a range based partitioning system, documents with “close” shard key values are likely to be in the same chunk, and therefore on the same shard

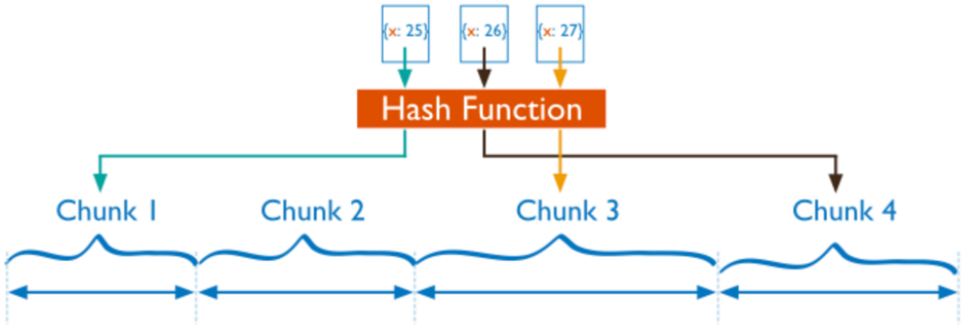


HASH BASED SHARDING

For hash based partitioning, MongoDB computes a hash of a field's value, and then uses these hashes to create chunks

With hash based partitioning, two documents with "close" shard key values are unlikely to be part of the same chunk.

This ensures a more random distribution of a collection in the cluster



TAG AWARE SHARDING

MongoDB allows administrators to direct the balancing policy using tag aware sharding.

Administrators create and associate tags with ranges of the shard key

Assign those tags to the shards.

The balancer migrates tagged data to the appropriate shards and ensures that the cluster always enforces the distribution of data that the tags describe

- Tags are the primary mechanism to control the behavior of the balancer and the distribution of chunks in a cluster.
- Most commonly, tag aware sharding serves to improve the locality of data for sharded clusters that span multiple data centers

RANGE & HASH BASED SHARDING

Range based partitioning supports more efficient range queries

- Given a range query on the shard key, the query router can easily determine which chunks overlap that range and route the query to only those shards that contain these chunks
- It can result in an uneven distribution of data, which may negate some of the benefits of sharding
  - If the shard key is a linearly increasing field, such as time, then all requests for a given time range will map to the same chunk, and thus the same shard.
  - In this situation, a small set of shards may receive the majority of requests and the system would not scale very well

Hash based partitioning, ensures an even distribution of data

- Hashed key values results in random distribution of data across chunks and therefore shards.
- Random distribution makes it more likely that a range query on the shard key will not be able to target a few shards but would more likely query every shard in order to return a result

MAINTAINING A BALANCED DISTRIBUTION

The addition of new data or the addition of new servers can result in data distribution imbalances within the cluster

- particular shard contains significantly more chunks than another shard
- a size of a chunk is significantly greater than other chunk sizes

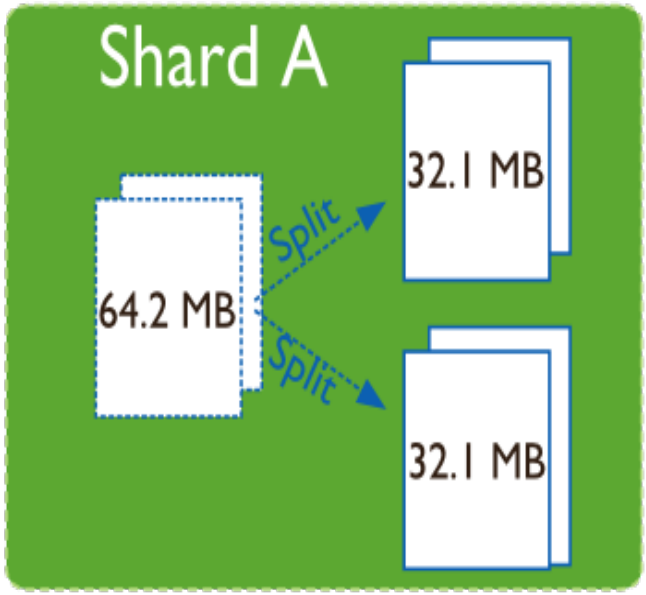
MongoDB ensures a balanced cluster using two background process: **splitting** and the **balancer**

SPLITTING

Background process that keeps chunks from growing too large.

When a chunk grows beyond a specified chunk size,

- MongoDB splits the chunk in half
- Inserts and updates triggers splits.
  - Splits are an efficient meta-data change.
  - To create splits, MongoDB does not migrate any data or affect the shards



BALANCING

The balancer is a background process that manages chunk migrations.

Runs from any of the query routers in a cluster

When the distribution of a sharded collection in a cluster is uneven

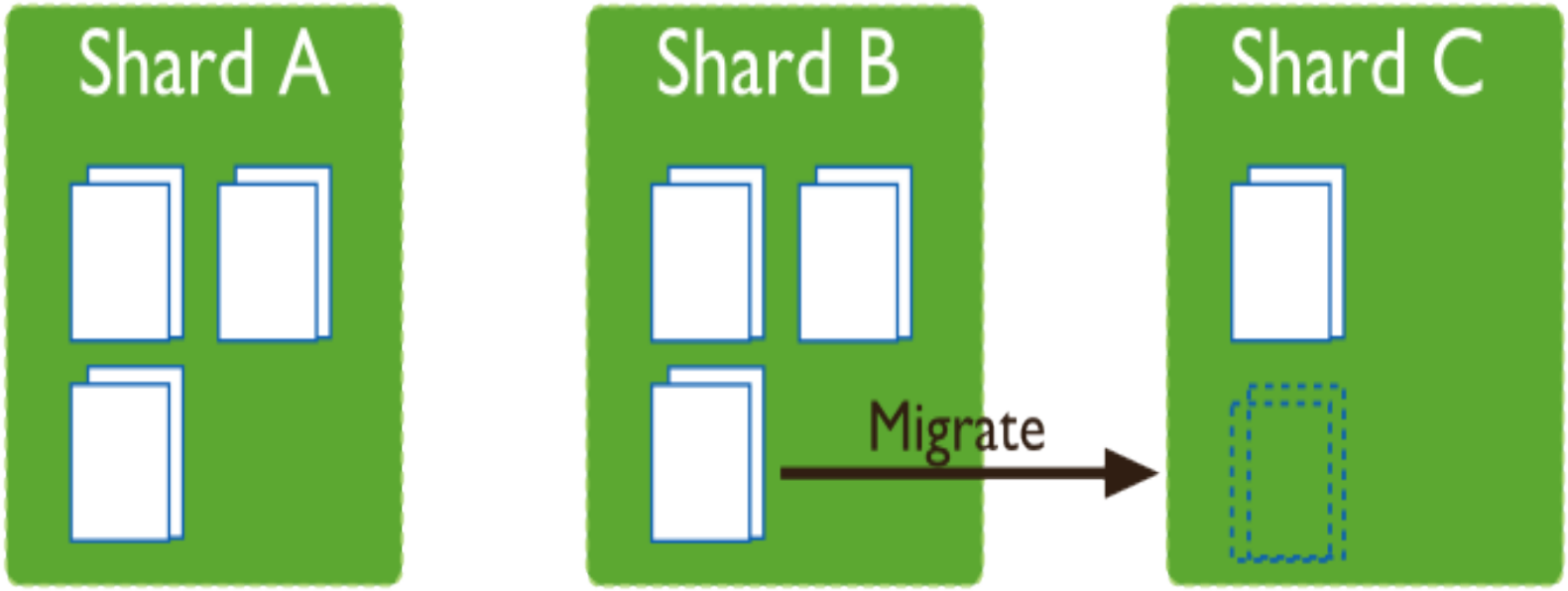
- The balancer process migrates chunks from the shard that has the largest number of chunks to the shard with the least number of chunks until the collection balances
- For example: if collection users has 100 chunks on shard 1 and 50 chunks on shard 2, the balancer will migrate chunks from shard 1 to shard 2 until the collection achieves balance

BALANCING (2)

The shards manage chunk migrations as a background operation between an origin shard and a destination shard.

- During a chunk migration, the destination shard is sent all the current documents in the chunk from the origin shard.
- Next, the destination shard captures and applies all changes made to the data during the migration process.
- Finally, the metadata regarding the location of the chunk on config server is updated
- If there's an error during the migration, the balancer aborts the process leaving the chunk unchanged on the origin shard.
- MongoDB removes the chunk's data from the origin shard after the migration completes successfully

BALANCING (3)



ADDING & REMOVING SHARDS FROM A CLUSTER

Adding a shard to a cluster creates an imbalance since the new shard has no chunks.

- MongoDB begins migrating data to the new shard immediately
- It can take some time before the cluster balances.

When removing a shard, the balancer migrates all chunks from a shard to other shards.

After migrating all data and updating the meta data, you can safely remove the shard

SHARDING VS HORIZONTAL PARITIONING

Horizontal partitioning splits one or more tables by row, usually within a single instance of a schema and a database server.

- Reduce index size (and thus search effort) provided that there is some obvious, robust, implicit way to identify in which table a particular row will be found, without first needing to search the index,
- e.g., the classic example of the '`CustomersEast`' and '`CustomersWest`' tables, where their zip code already indicates where they will be found

Sharding goes beyond this: it partitions the problematic table(s) in the same way, but across potentially multiple instances of the schema

- Search load for the large partitioned table can now be split across multiple servers (logical or physical), not just multiple indexes on the same logical server
- Splitting shards across multiple isolated instances requires more than simple horizontal partitioning
- Sharding splits large partitionable tables across the servers, while smaller tables ***are replicated as complete units***


hands
on

Genoveva Vargas-Solar

CR1, CNRS, LIG-LAFMIA

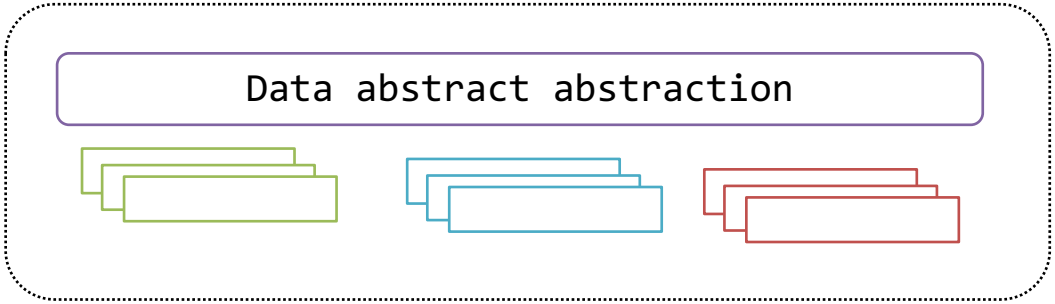
Genoveva.Vargas@imag.fr

http://vargas-solar.com/datascience

Polyglot Persistence

BUffer

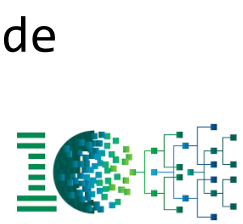
Replacement strategy
Data organization: data
Co-locataion, sharding and replication



Transformation files → pivot data abstraction

Transformation pages → records

Node



Buffer management:
Replacement strategy