ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

INTRODUCTION TO NOSQL DATABASES

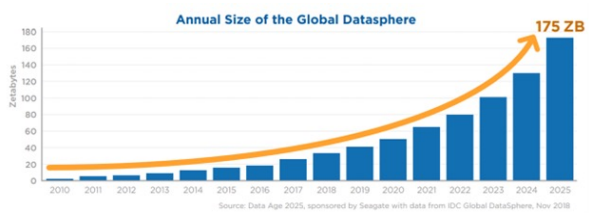
Nicolas.travers@devinci.fr

THE BIG DATA CONTEXT

Applications and Web platforms

- Exponential growth of the amount of Data x2 / 2 years
- Unprecedent management of this volume

- Need distribution: Computation & Data
- Clusters: Huge number of servers (Google, Amazon, Facebook, etc.)
  - DataCenter : ~5000 servers/data center
  - Google : ~1M de servers



IDC predicts that the Global Datasphere will grow from **33 Zettabytes** in 2018 to **175 Zettabytes** by 2025

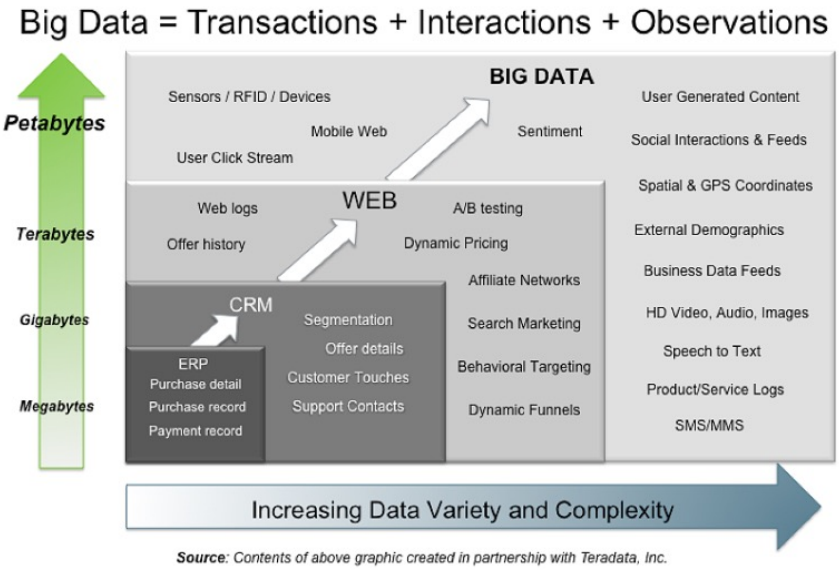
DBMS VS THE PROBLEM OF 3V

- **Volume**
  - Designed to store GB/TB
  - But needs PB (maybe EB)
- **Variety**
  - Heterogeneous data
  - Variable types
  - Text, semi-structured
- **Velocity**
  - Fast data production

Eventually
- **Value**
- **Veracity**

ESILV
Introduction to NoSQL

DBMS VS THE PROBLEM OF 3V



***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

DBMS VS DISTRIBUTION

- Pros DBMS
  - Joins between tables
  - SQL: Rich query language
  - Constraints: integrity, data types, normal forms
- Cons for Distribution
  - How to partition/place data?
  - How make joins?
  - How to manage fault tolerance?

ACID VS BASE

Local DBMS: **ACID** transactions

- **A**tomicity: integral completion or none
- **C**onsistency: consistent at start and end
- **I**solation: no communication between them
- **D**urability: an operation cannot be reversed

Distributed systems: **BASE**

- ***B**asically **A**vailable* :
  - Any request => An answer
  - Even in a changing state
- ***S**oft-state* :
  - Opposite to Durability.
  - System's state (servers or data) could change over time (without any update)
- ***E**ventually consistent* :
  - With time, data can be consistent
  - Updates have to be propagated

WHAT IS NOSQL?

NoSQL : Not Only SQL

- New data storage/management approach
- Scales up the system (through distribution)
- Fault tolerant architecture (replication)
- Complex metadata management
  - Schema-Last approach

Do not substitute DBMS!!

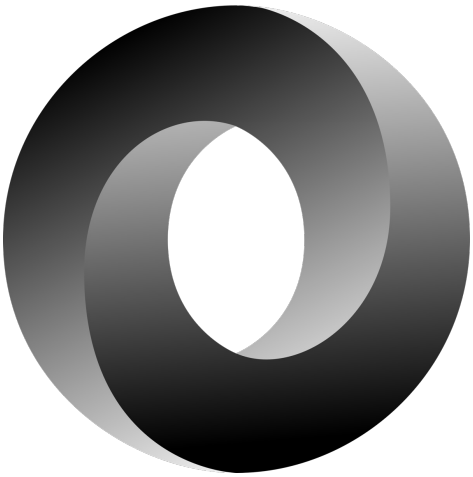
NoSQL's target:

- Schema-first with SQL
- Very huge volume of data (PetaBytes)
- Very short response time
- Consistency is not mandatory

nosql

NOSQL CHARACTERISTICS

- No more relations
  - No schema (hardly data types)
  - Dynamic schema
  - ➢ Collections
- Schema-Last
  - No more tuples (1st & *3rd Normal Forms)*
  - Nesting
  - Arrays
  - ➢ JSON documents
- Distribution
  - Parallelism (abstraction of Map/Reduce, not cloud computing)
- Data replication
  - Availability vs Consistency
  - No more transactions
  - ➢ Few writes, many reads

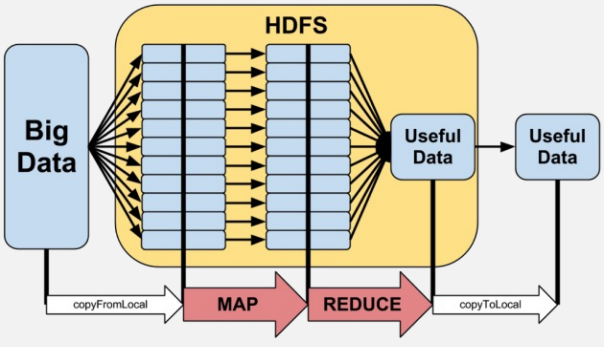


SHARDING: DATA DISTRIBUTION & REPLICATION STRATEGIES

- Files are segmented in chunks (64MB, 256MB…)
  - Chunks are distributed in the cluster
  - Data are placed according to a **sharding** strategy

- 3 types of technics of sharding:
  1. HDFS: Resource allocation based (racks, switches, datacenter)
     - *++ Fault tolerance and massive computations*
  2. *Clustered index:* Tree-based structure (total order)
     - *++ Physical data clustering and dynamicity*
  3. DHT: Hash-based structure
     - *++ elasticity and self-management*

SHARDING WITH HDFS

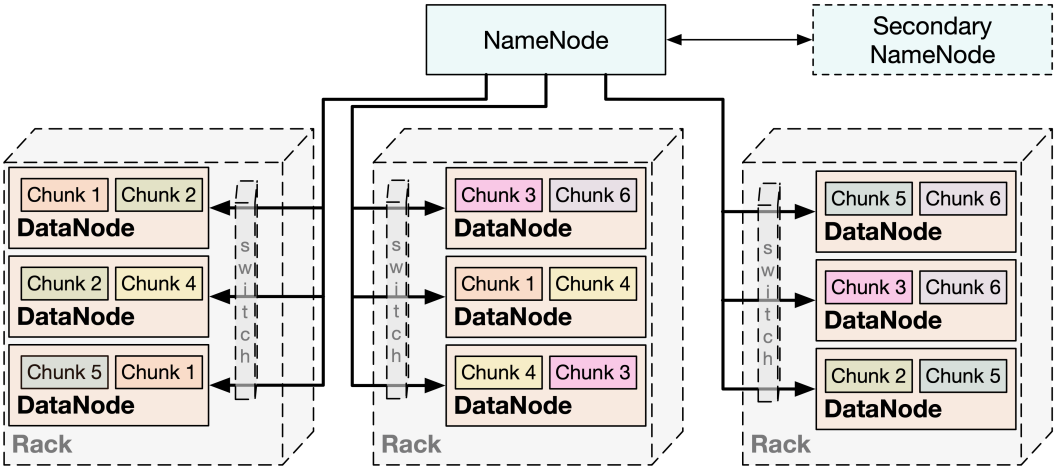


Resources allocation strategy

- Distributed file system
- Rely on servers load balancing
- Dedicated to fault tolerance
- Dynamic allocation and optimization

(1) See *Hadoop Distributed File System*
http://chewbii.com/transparents-hdfs-hadoop/

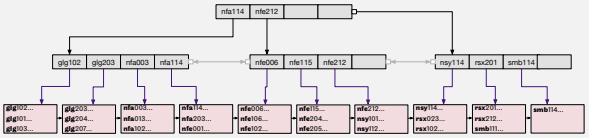
SHARDING WITH HDFS: EXAMPLE



SHARDING WITH HDFS NOSQL SOLUTIONS

Highly distributed computation

Fault Tolerance



SHARDING WITH CLUSTERED INDEX

Distributed Clustered Index

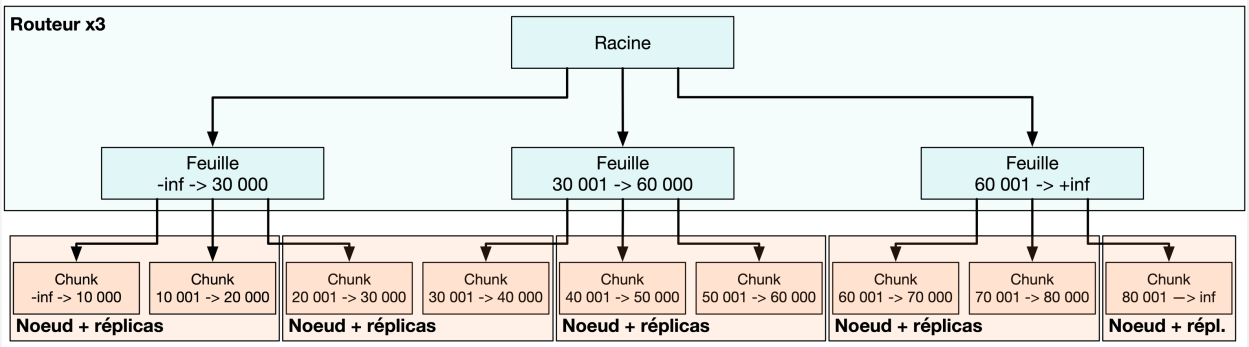
- Sorted data according to a key
  - Primary key
  - Build chunks (default 256Mo)
  - Distribute chunks
  - Replicate chunks

➡️ *Range* queries and *group by* queries

⚠️ Only one key can be chosen

(2) See « index non-dense »
http://chewbii.com/videos-optimisation-bases-de-donnees/
(Vidéo 4)

SHARDING WITH CLUSTERED INDEX EXAMPLE



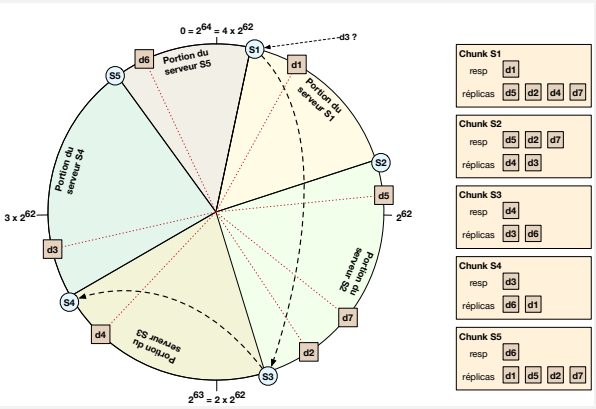
SHARDING WITH CLUSTERED INDEX SOLUTIONS

Data organization

Complex queries

mongoDB.

SHARDING WITH DHT

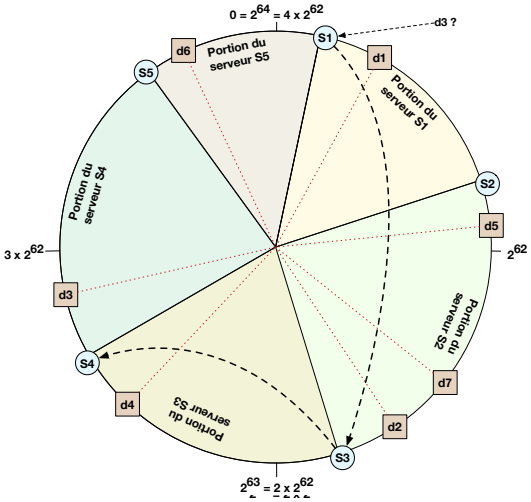


Distributed Hash Table (DHT[3])

- Ring of virtual servers
  - Max 2^{64}
- Unique hash table: splitted & distributed
  - Routing
  - Efficiency
  - Self-Management (no main server)

(3) See DHT http://chewbii.com/hachagedynamique/
(Vidéo 4 & 5)

SHARDING WITH DHT: EXAMPLE



Chunk S1					
resp	d1				
réplicas	d5	d2	d4	d7	

Chunk S2					
resp	d5	d2	d7		
réplicas	d4	d3			

Chunk S3					
resp	d4				
réplicas	d3	d6			

Chunk S4					
resp	d3				
réplicas	d6	d1			

Chunk S5					
resp	d6				
réplicas	d1	d5	d2	d7	

SHARDING WITH DHT: SOLUTIONS

Self-Management

Elasticity

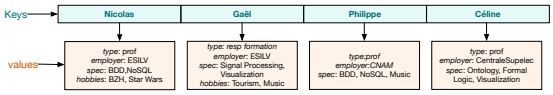
cassandra

redis

DynamoDB

NOSQL DATA TYPE FAMILIES

Key-Value stores



Wide Column-oriented

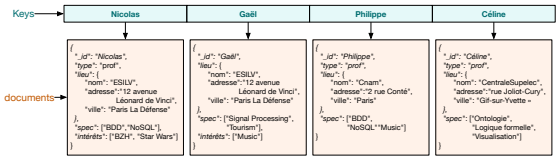
id	status
Nicolas	prof
Gaël	resp formation
Philippe	prof
Céline	prof

id	employer
Nicolas	ESILV
Gaël	ESILV
Philippe	CNAM
Céline	CentraleSupelec

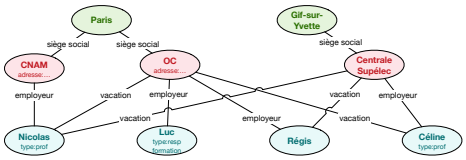
id	spec
Nicolas	BDD
Nicolas	NoSQL
Gaël	Signal Processing
Gaël	Visualization
Philippe	BDD
Philippe	NoSQL
Philippe	Music
Céline	Ontology
Céline	Formal Logic
Céline	Visualization

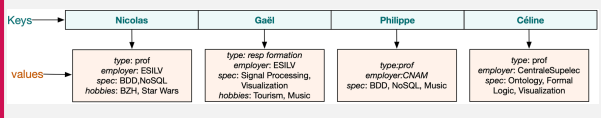
id	hobbies
Nicolas	BZH
Nicolas	Star Wars
Gaël	Tourism
Gaël	Music

Document-oriented



Graph oriented





KEY-VALUE STORE

Similar to a distributed "*HashMap*"

Key + Value:

- No fixed schema on values (strings, object, integer, binaries…)

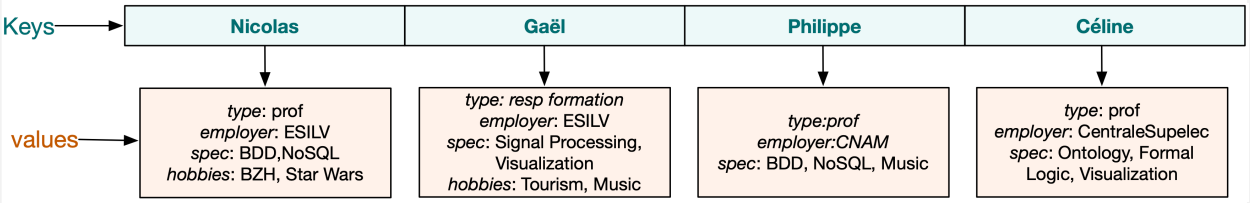
Drawbacks:

- **No structures** nor typing
- No structured-based queries

KEY-VALUE STORE: EXAMPLE

Relational database

id	status	employer	spec	hobbies
Nicolas	prof	ESILV	BDD, NoSQL	BZH, Star Wars
Gaël	resp formation	ESILV	Signal Processing, Visualization	Tourism, Music
Philippe	prof	CNAM	BDD, NoSQL, Music	
Céline	prof	CentraleSupelec	Ontology, Formal Logic, Visualization	



KEY-VALUE STORE: QUERIES

CRUD

- CREATE (**key**, value)
  - CREATE ("Nicolas", "type:'prof',employer:'ESILV',spec:'BDD,NoSQL',hobbies:'BZH,Star Wars' ") → OK
- READ(**key**)
  - READ("Nicolas") → "type:'prof',employer:'ESILV',spec:'BDD,NoSQL',hobbies:'BZH,Star Wars' "
- UPDATE(**key**, value)
  - UPDATE("Nicolas", "type:'prof',employer:ESILV',spec:'BDD,NoSQL' ") → OK
- DELETE(**key**)
  - DELETE("Nicolas") → OK

KEY-VALUE STORE: USE CASES

- Web site logs store
- Web site/DB cache
- User profil management
- Sensors status
- E-commerce bucket storage

KEY-VALUE STORE: SOLUTIONS

Easy to set up

Efficiency

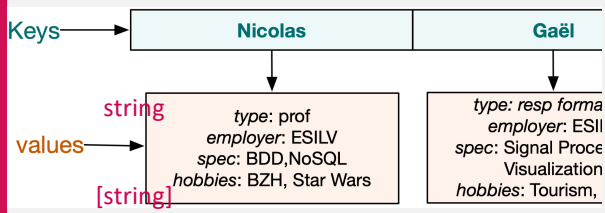
Microsoft Azure

MEMCACHED

redis

Amazon SimpleDB

WIDE COLUMN-ORIENTED STORE



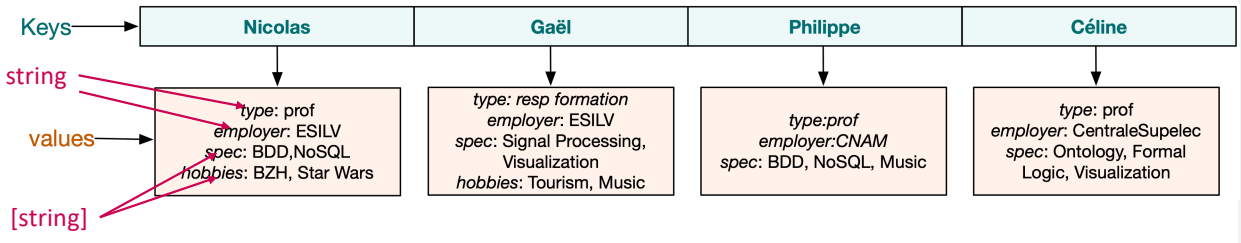
Wide Column ≠ Columnar ≠ Column

- **Wide-Column** -> tuples with enhanced typing (arrays, dictionaries, dynamic schema)
  - Remain DBMS oriented with strong typing (query validation)
  - Also called wide-column oriented stores
- **Columnar** -> 1 attribute = 1 storage **file**
  - Column-oriented: attributes are separated in column-families and **stored independently**
  - Efficient queries on columns (**aggregates**)
  - Not NoSQL -> In Memory databases
- **Column** -> highly structured tuples = relational (SQL)

WIDE COLUMN-ORIENTED STORE: EXAMPLE

Relational database

id	status	employer	spec	hobbies
Nicolas	prof	ESILV	BDD, NoSQL	BZH, Star Wars
Gaël	resp formation	ESILV	Signal Processing, Visualization	Tourism, Music
Philippe	prof	CNAM	BDD, NoSQL, Music	
Céline	prof	CentraleSupelec	Ontology, Formal Logic, Visualization	



WIDE COLUMN-ORIENTED STORES ≠ COLUMNAR ORIENTED: EXAMPLE

id	status	employer	spec	hobbies
Nicolas	prof	ESILV	BDD, NoSQL	BZH, Star Wars
Gaël	resp formation	ESILV	Signal Processing, Visualization	Tourism, Music
Philippe	prof	CNAM	BDD, NoSQL, Music	
Céline	prof	CentraleSupelec	Ontology, Formal Logic, Visualization	

id	status
Nicolas	prof
Gaël	resp formation
Philippe	prof
Céline	prof

id	employer
Nicolas	ESILV
Gaël	ESILV
Philippe	CNAM
Céline	CentraleSupelec

id	spec
Nicolas	BDD
Nicolas	NoSQL
Gaël	Signal Processing
Gaël	Visualization
Philippe	BDD
Philippe	NoSQL
Philippe	Music
Céline	Ontology
Céline	Formal Logic
Céline	Visualization

id	hobbies
Nicolas	BZH
Nicolas	Star Wars
Gaël	Tourism
Gaël	Music

WIDE COLUMN-ORIENTED STORES USE CASES

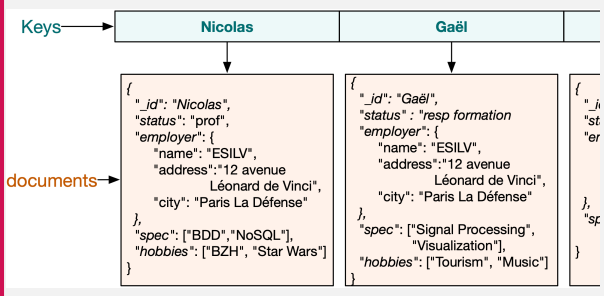
- Statistics on online polls
- Logging
- Category searchs on products (ie. Ebay)
- Reporting

WIDE COLUMN-ORIENTED STORE: SOLUTIONS

Strong typing

Simple queries

Google
Big Query



DOCUMENT-ORIENTED STORES

Based on the key-value store

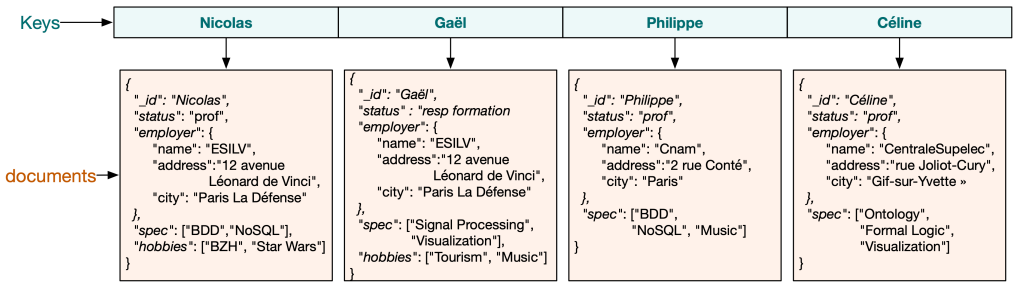
- Add semi-structured data (JSON/XML)
- Keys indexing
- Lists of values, nesting
- Allow fusion of concepts

HTTP API

- More complex than CRUD
- Rich queries (database)

DOCUMENT-ORIENTED STORES: EXAMPLE

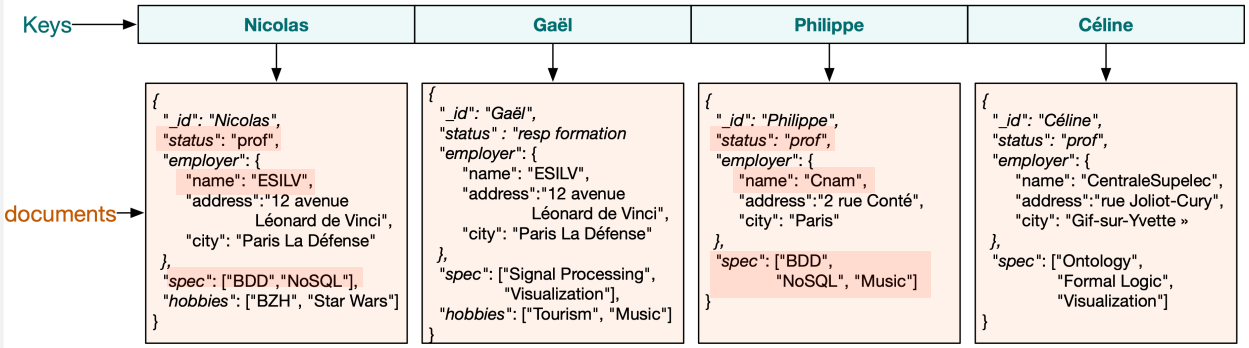
id	status	employer	spec	hobbies
Nicolas	prof	ESILV	BDD, NoSQL	BZH, Star Wars
Gaël	resp formation	ESILV	Signal Processing, Visualization	Tourism, Music
Philippe	prof	CNAM	BDD, NoSQL, Music	
Céline	prof	CentraleSupelec	Ontology, Formal Logic, Visualization	



DOCUMENT-ORIENTED STORES: QUERIES

Manipulations on documents content

- Establishment (employer.name) of professors (status) specialized in BDD (*in* spec)



DOCUMENT-ORIENTED STORES: USE CASES

- CMS: Content Management Systems
  - Online libraries
  - Products management
  - Software logging
  - Metadata on multimedia stores
- Collection of complex events
- Emails storage
- Social networks logging

DOCUMENT-ORIENTED STORES: SOLUTIONS

Richness of queries

Manage objects

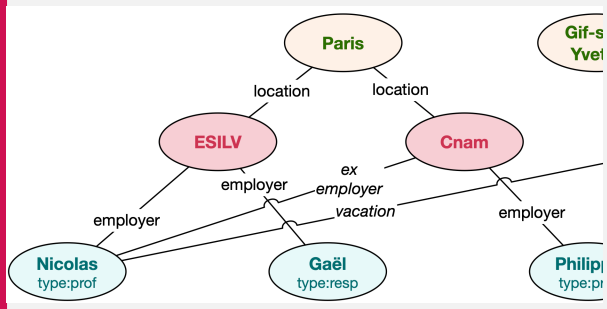
Couchbase

DynamoDB

mongoDB®

elasticsearch

GRAPH-ORIENTED STORES

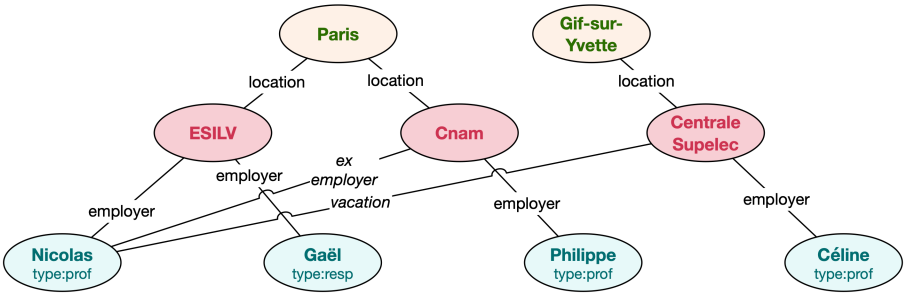


Storage: nodes, relations and properties

- Graph Theory
- **Pattern queries** on the graph
- Data are loaded on demand
- Difficulties for **data modeling**

GRAPH-ORIENTED STORES: EXAMPLE

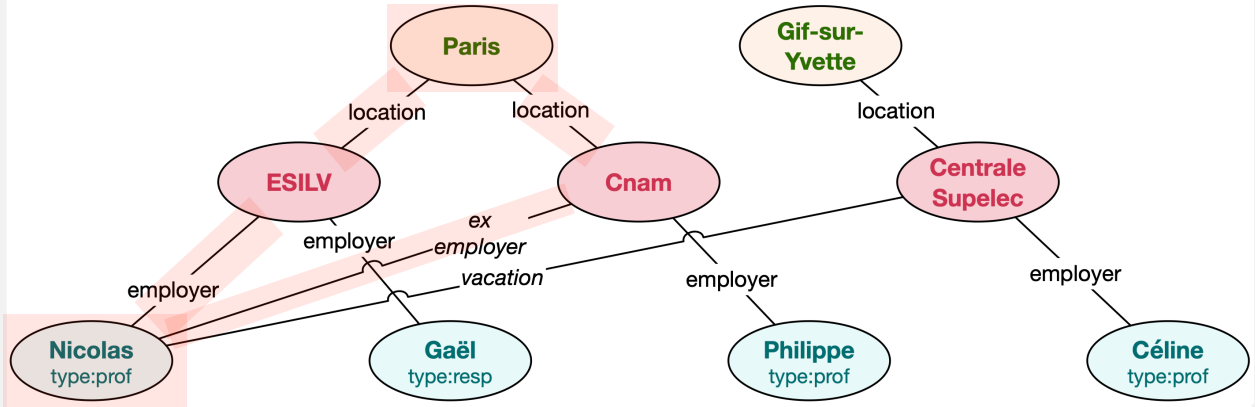
id	status	employer	spec	hobbies
Nicolas	prof	ESILV	BDD, NoSQL	BZH, Star Wars
Gaël	resp formation	ESILV	Signal Processing, Visualization	Tourism, Music
Philippe	prof	CNAM	BDD, NoSQL, Music	
Céline	prof	CentraleSupelec	Ontology, Formal Logic, Visualization	



GRAPH-ORIENTED STORES: QUERIES

Pattern queries

- **Persons** who **had 2 employers** at **Paris**



GRAPH-ORIENTED STORES: USE CASES

- Pattern recognition
  - Recommandation
  - Fraud detection
  - SIG (road network, electricty, reachability)
- Graph computations
  - Shorted path
  - PageRank
  - Connexity
  - Communities detection
- Linked data

GRAPH-ORIENTED STORES: SOLUTIONS

Recommendation

Network

Knowledge Management

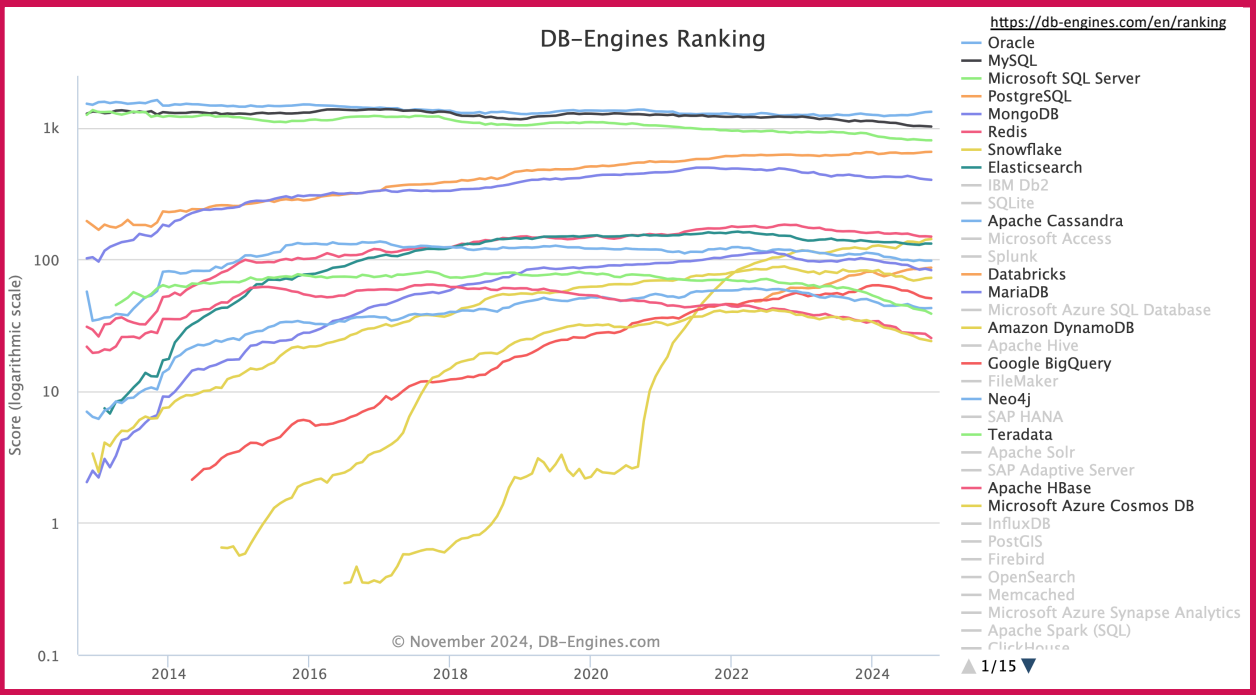
OrientDB®

Azure Cosmos DB:

neo4j

FlockDB

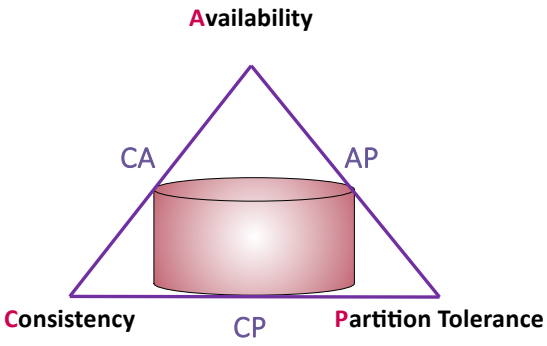
DATASTAX ENTERPRISE



THE CAP THEOREM [BREWER 2000]

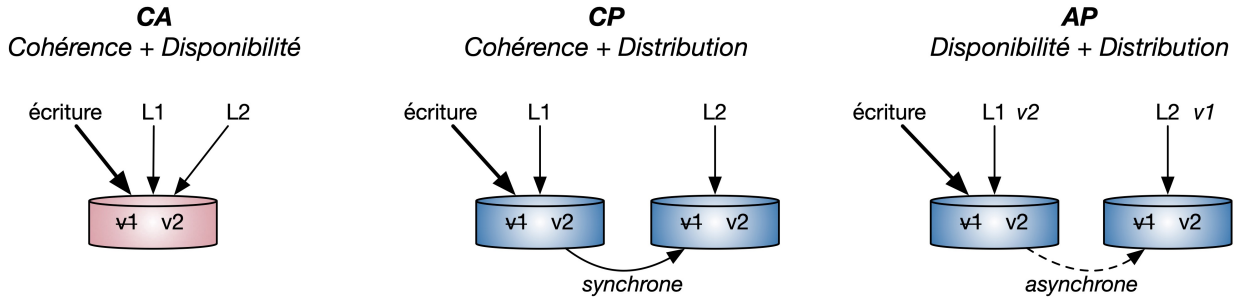
3 main properties for distributed management

1. *Consistency*:
   - A data returns the proper value at any time (coherency)
2. *Availability*:
   - Even if a server is down, data is available
3. *Partition Tolerance*:
   - Even if the system is partitioned, a query must have an answer (unless for global failures)

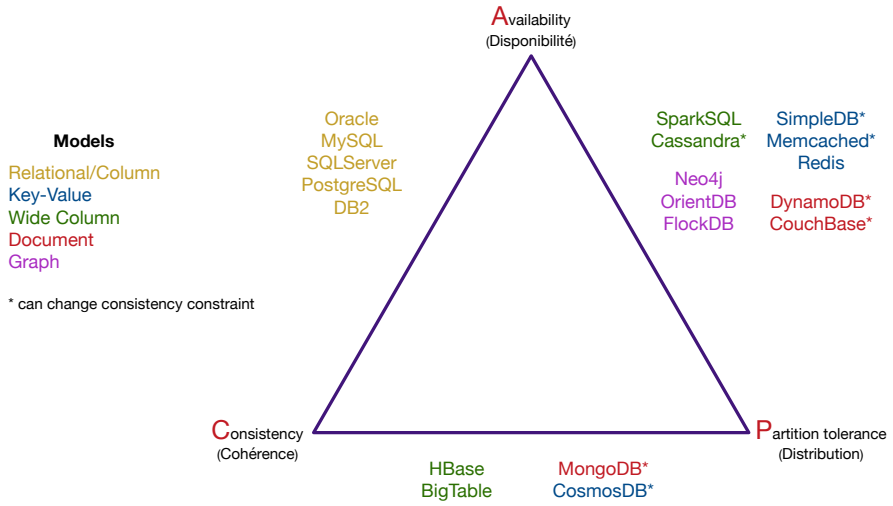


Theorem: *A distributed, networked system can have only <u>two</u> of these three properties.*

THE CAP THEOREM: ILLUSTRATION



THE CAP THEOREM: CAP TRIANGLE



CAP VS PACELC [ABADI 2012]

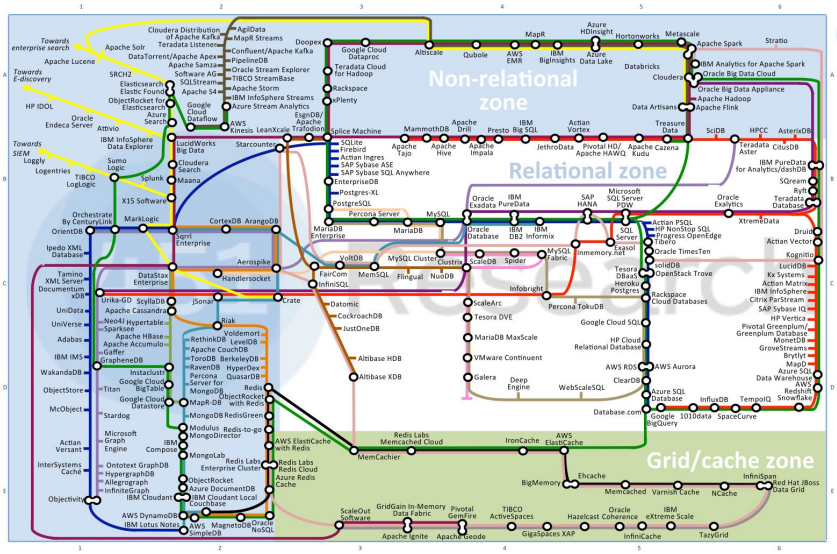
Data synchronization

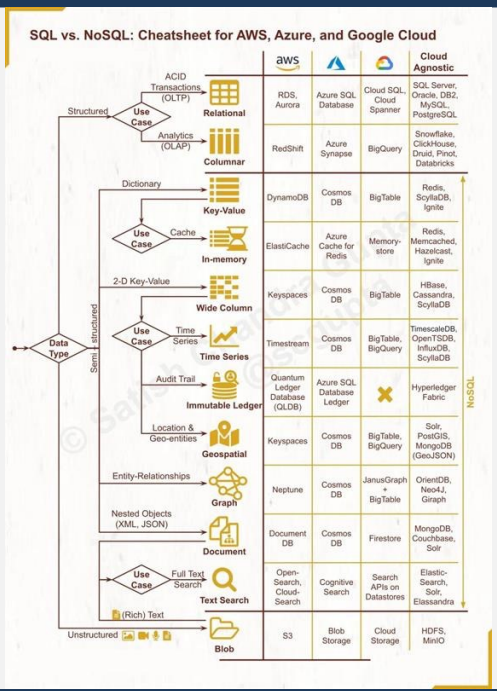
- PAC
  - Partitioning/Data movements
  - Between Availability and Consistency
- ELC
  - Normal state of the network
  - Between Latency and Consistency

- **PA/EL**
  - *Dynamo, Cassandra, Riak*
  - Always **available** with a minimum of latency
- **PC/EC**
  - *VoltDB/H-Store, MegaStore*
  - **Consistency** whatever happens
- **PA/EC**
  - *MongoDB*
  - While partitioning: **availability**
  - In normal state: **consistency**

DATABASE CHOICE?

- Data Model
- Sharding strategy
- Consistency vs Efficiency
- Community
  +
- Change management
- Transformation cost





DATABASE CHOICE

Credits - *Satish Chandra Gupta*

MAP/REDUCE (SHORTLY)

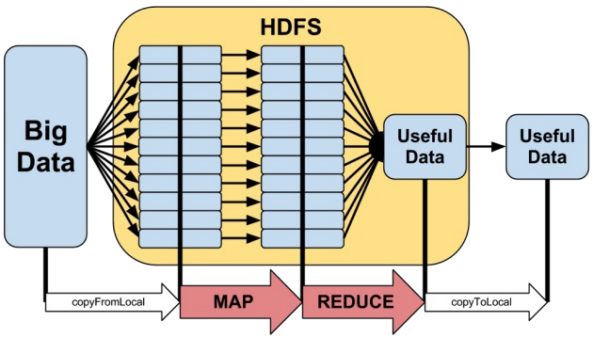
Distributed computation framework

2 main functions

- ***Map***: data transformation
  - input: 1 data
  - output: several pairs (**key/value**)
- ***Reduce***: aggregate values for each key
  - *input : values for a given key :* ***key + list(values)***
  - *output : 1 value* **key + value**

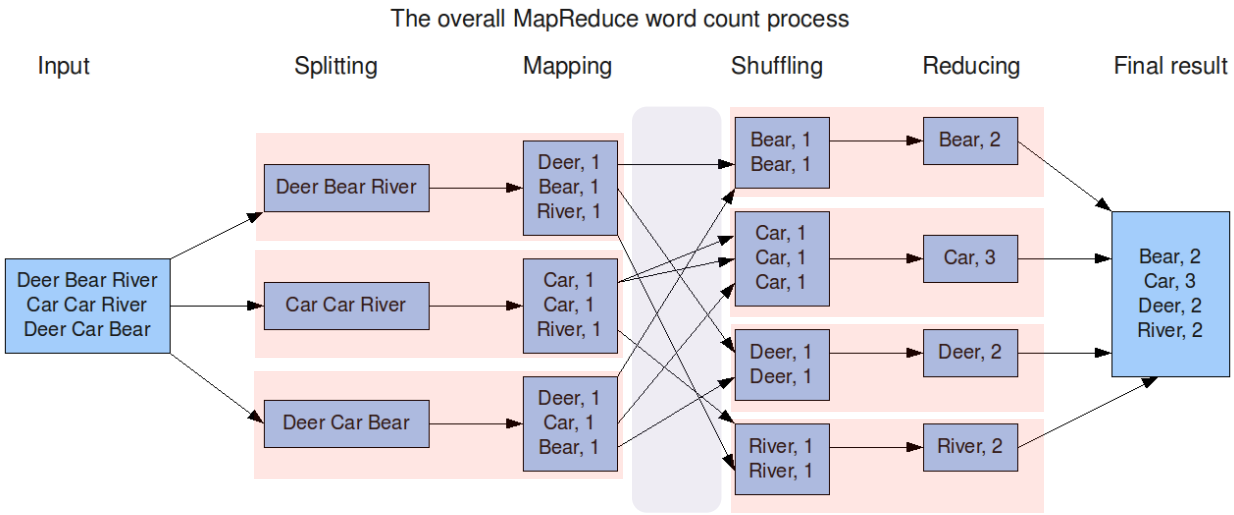
Scalability, Fault Tolerance

- *Send map&reduce to the cluster (jobs)*
- *In case of failure of 1 job,*
  restart the part of the job on another server



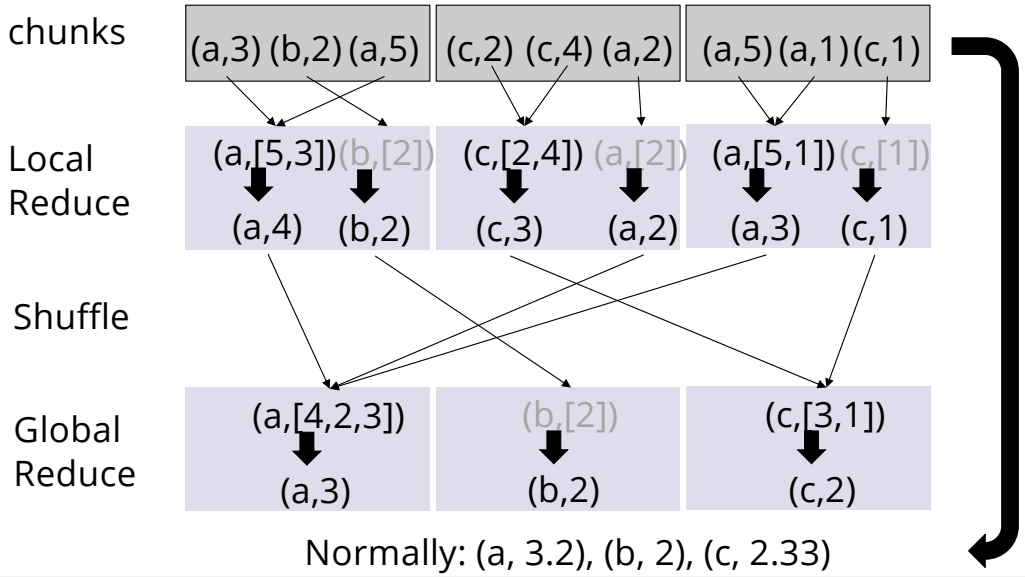
ESILV
Introduction to NoSQL

MAPREDUCE: WORD COUNT EXECUTION



LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr
Nicolas.travers@devinci.fr

REDUCE: AVERAGE FUNCTION (1/2)



MAPREDUCE: PROPERTIES

Programming language

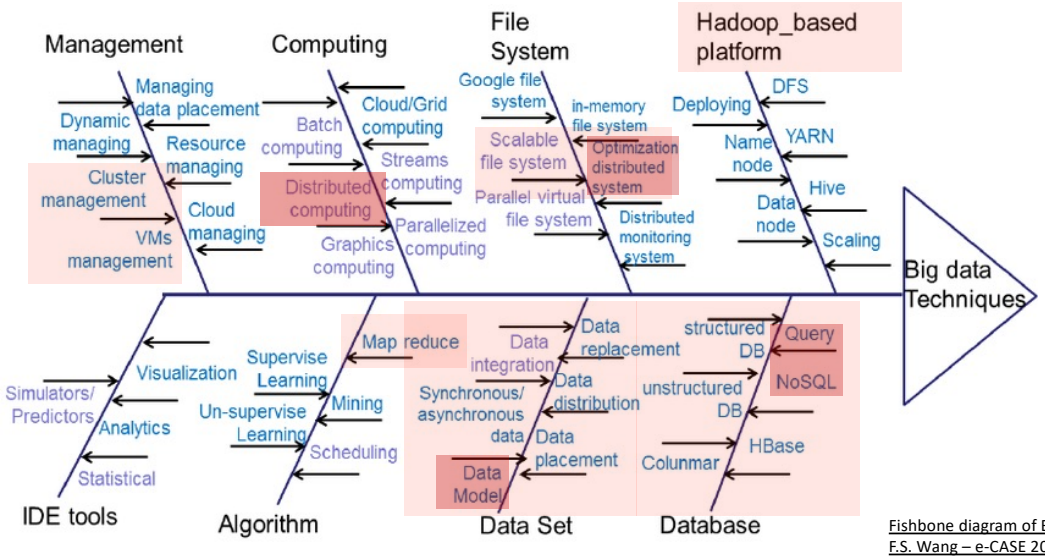
- Highly distributable
- Fault tolerant
- Simple to express

- ***NoSQL*** -> on-top of MapReduce (operations are developped in MR)

Drawbacks

- No join queries
- Many networks communications (shuffle)

BIG DATA OVERVIEW – COURSE'S FOCUS



Fishbone diagram of Big Data
F.S. Wang – e-CASE 2014