



INTRODUCTION TO NOSQL DATABASES

Nicolas.travers@devinci.fr



Introduction to NoSQL

THE BIG DATA CONTEXT

Applications and Web platforms

- Exponential growth of the amount of Data **x2 / 2 years**
- Unprecedented 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

Data Never Sleeps 11.0

DOMO has been keeping tabs on the world's data usage—in a minute—for over a decade now. What the numbers consistently show is that how we use data is always evolving—and that data isn't slowing down, even as we work to manage it better. The rise of artificial intelligence (AI), the move to the new web, content, work, and create, digital payments continue to replace traditional transactions, Taylor Swift streams in countries worldwide, and a host of other new games and apps fuel digital experiences.

In Domo's 11th edition of Data Never Sleeps, we take the pulse of our digital age, where every click, swipe, and stream feeds an ever-expanding digital universe. These are not just numbers; they are the heartbeat of a world where data never sleeps.

Platform/Service	Usage/Action
AIRBNB	74.7 STAYS
AMAZON	\$455K SPENDING PER MINUTE
X	360K TWEETS
GOOGLE	6.3M SEARCHES PER MINUTE
WHATSAPP	41.6M MESSAGES PER MINUTE
LINKEDIN	6,060 MESSAGES PER MINUTE
CYBER-CRIMINALS	30 BOSS ATTACKS LAUNCHED PER MINUTE
INSTAGRAM	6,944 PHOTOS UPLOADED PER MINUTE
CHATGPT	6,944 PROMPTS PER MINUTE
FACEBOOK	4M POSTS PER MINUTE
TAYLOR SWIFT	69.4K STREAMS PER MINUTE
THE AVERAGE PERSON	102 MB OF DATA PER MINUTE
PEOPLE TRADE BONDS	\$398M PER MINUTE
EMAILS	241M PER MINUTE
TWITCH	48K HOURS OF CONTENT PER MINUTE
DOORDASH	\$122K IN ORDERS PER MINUTE
INSTAGRAM	6,948 REELS VIA DM PER MINUTE
VENMO	\$463K IN TRANSACTIONS PER MINUTE

EVERY MINUTE 01:00 OF THE DAY

Global Internet Population Growth (in billions)

Year	Population (billions)
2013	~7.5
2016	~8.0
2020	~8.5
2023	~9.0

Learn more at [domo.com](https://www.domo.com)

DBMS VS THE PROBLEM OF 3V

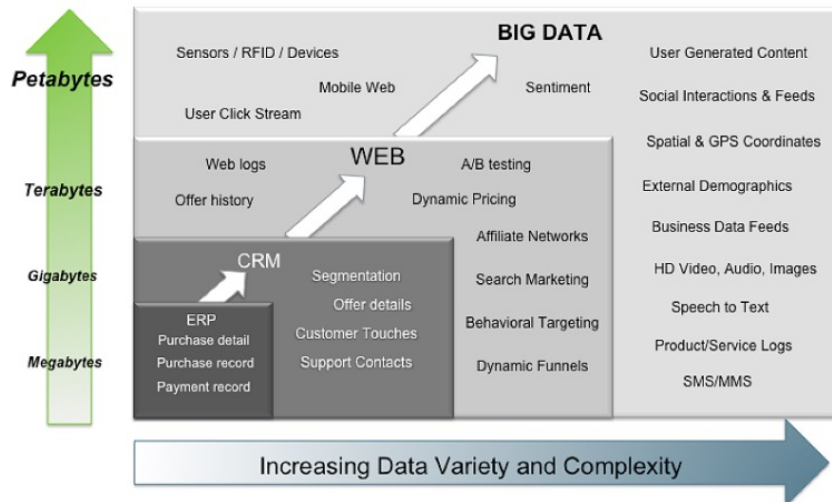
- **Volume**
Designed to store GB/TB
But needs PB (maybe EB)
- **Variety**
Heterogeneous data
Variable types
Text, semi-structured
- **Velocity**
Fast data production

Eventually

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

DBMS VS THE PROBLEM OF 3V

Big Data = Transactions + Interactions + Observations



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

- **Atomicity:** integral completion or none
- **Consistency:** consistent at start and end
- **Isolation:** no communication between them
- **Durability:** an operation cannot be reversed

Distributed systems: BASE

- **Basically Available :**
 - Any request => An answer
 - Even in a changing state
- **Soft-state :**
 - Opposite to Durability.
 - System's state (servers or data) could change over time (without any update)
- **Eventually 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!!

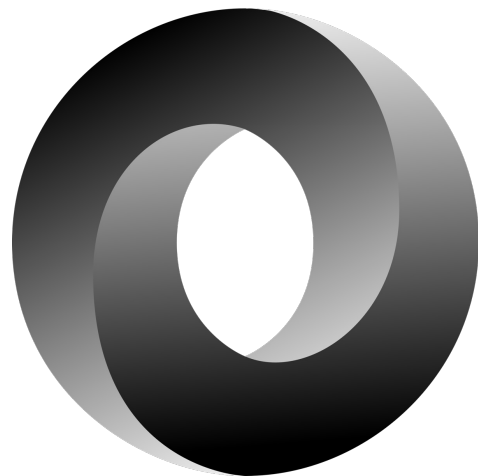
NoSQL's target:

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



NOSQL CHARACTERISTICS

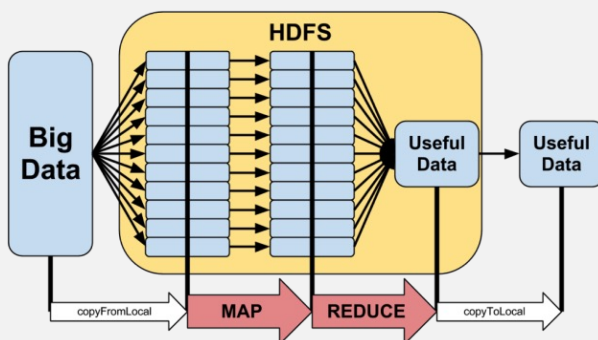
- **No more relations**
 - No schema (hardly data types)
 - Dynamic schema
 - Collections
- **Schema-Last**
 - No more tuples (1st & 3rd Normal Forms)
 - Nesting
 - Arrays
 - JSON documents
- **Distribution**
 - Parallelism (abstraction of Map/Reduce, not cloud computing)
- **Data replication**
 - Availability 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:
 - HDFS**: Resource allocation based (racks, switches, datacenter)
 - ++ *Fault tolerance and massive computations*
 - Clustered index**: Tree-based structure (total order)
 - ++ *Physical data clustering and dynamicity*
 - DHT**: Hash-based structure
 - ++ *elasticity and self-management*

SHARDING WITH HDFS

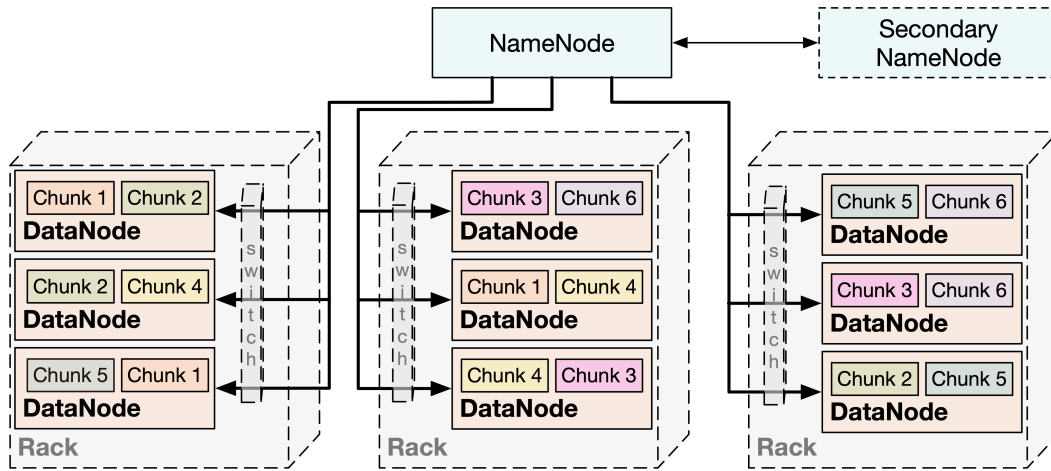


Resources allocation strategy

- 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

APACHE
HBASE

Spark  SQL



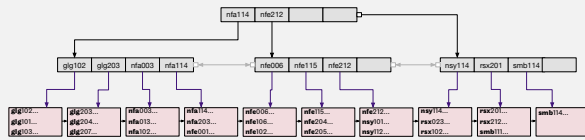
SHARDING WITH CLUSTERED INDEX

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

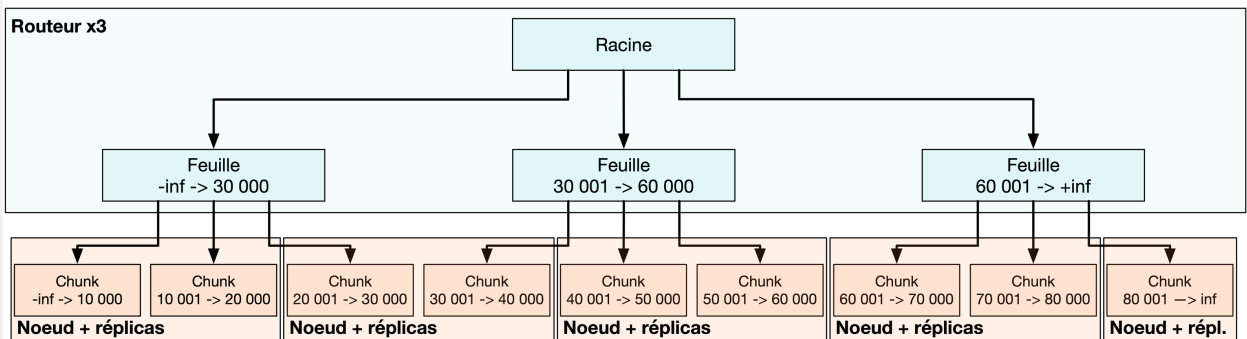


(2) See « index non-dense »

<http://chewbii.com/videos-optimisation-bases-de-donnees/>
(Vidéo 4)

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

SHARDING WITH CLUSTERED INDEX EXAMPLE



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

SHARDING WITH CLUSTERED INDEX SOLUTIONS

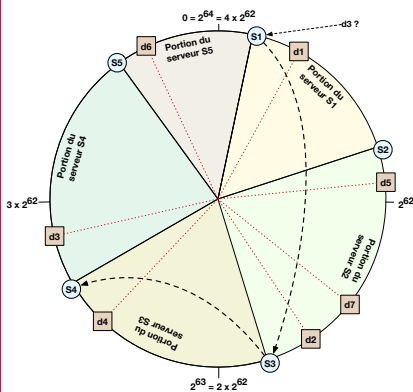
Data organization
Complex queries



SHARDING WITH DHT

Distributed Hash Table (DHT⁽³⁾)

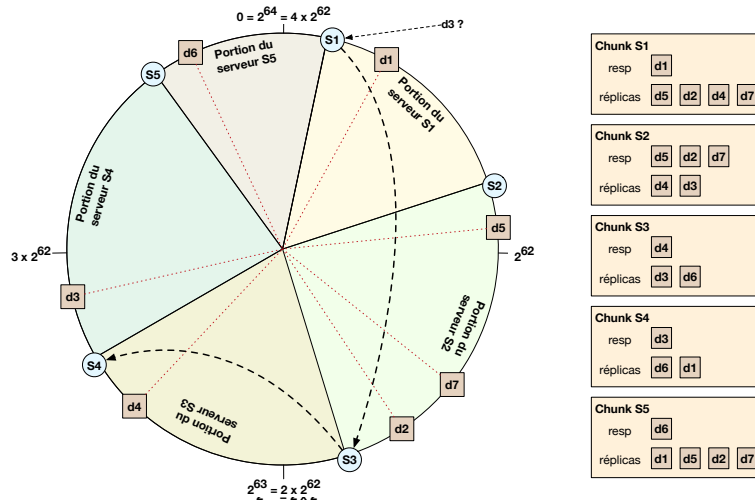
- Ring of virtual servers
 - Max 2^{64}
- Unique hash table: splitted & distributed
 - Routing
 - Efficiency
 - Self-Management (no main server)



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

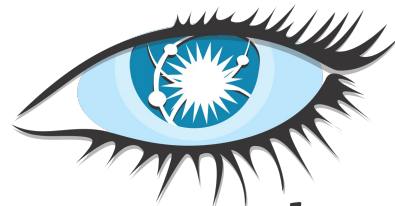
(3) See DHT <http://chewbii.com/hachagedynamique/>
(Vidéo 4 & 5)

SHARDING WITH DHT: EXAMPLE

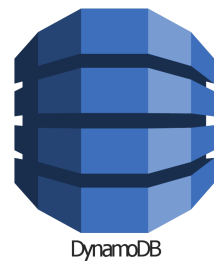


SHARDING WITH DHT: SOLUTIONS

Self-Management
Elasticity

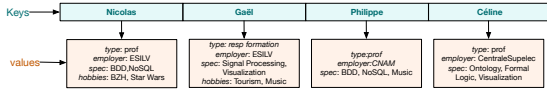


cassandra



NOSQL DATA TYPE FAMILIES

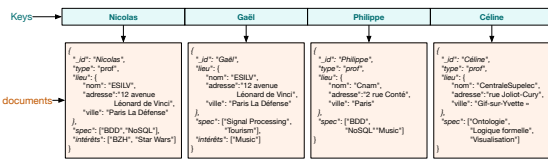
Key-Value stores



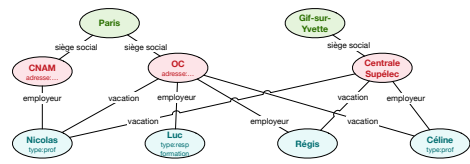
Wide Column-oriented

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

Document-oriented



Graph oriented



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

KEY-VALUE STORE

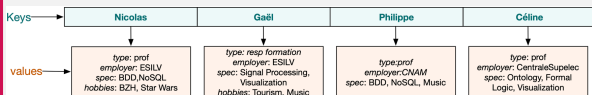
Similar to a distributed "HashMap"

Key + Value:

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

Drawbacks:

- No structures nor typing
- No structured-based queries

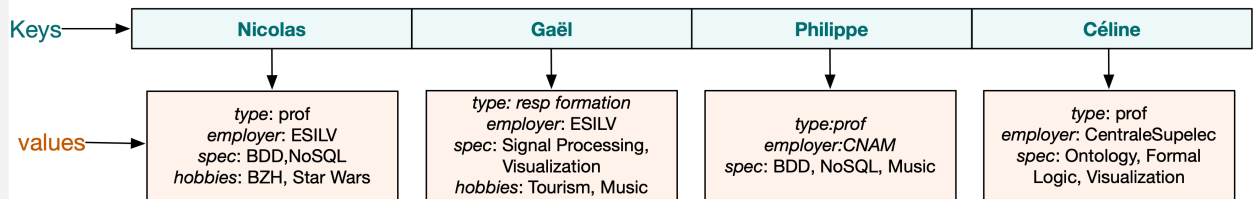


LIKE, SHARE, FOLLOW ! @esiv_paris | esilv.fr
Nicolas.travers@devinci.fr

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	CentraleSupélec	Ontology, Formal Logic, Visualization	



KEY-VALUE STORE: QUERIES

CRUD

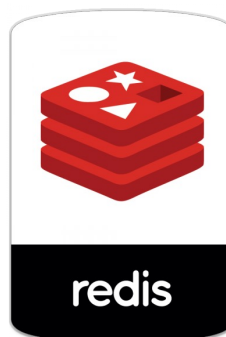
- CREATE (**key**, **value**)
 - CREATE ("Nicolas", {"type:'prof',employer:'ESILV',spec:'BDD,NoSQL',hobbies:'BZH,Star Wars' "}) → OK
- READ(**key**)
 - READ("Nicolas") → {"type:'prof',employer:'ESILV',spec:'BDD,NoSQL',hobbies:'BZH,Star Wars' "}
- UPDATE(**key**, **value**)
 - UPDATE("Nicolas", {"type:'prof',employer:'ESILV',spec:'BDD,NoSQL'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

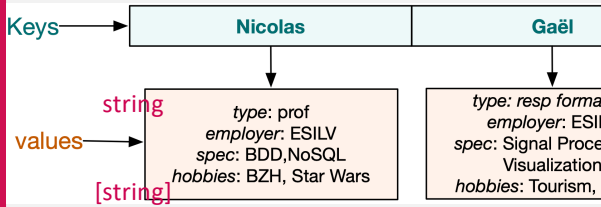


Amazon SimpleDB

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)

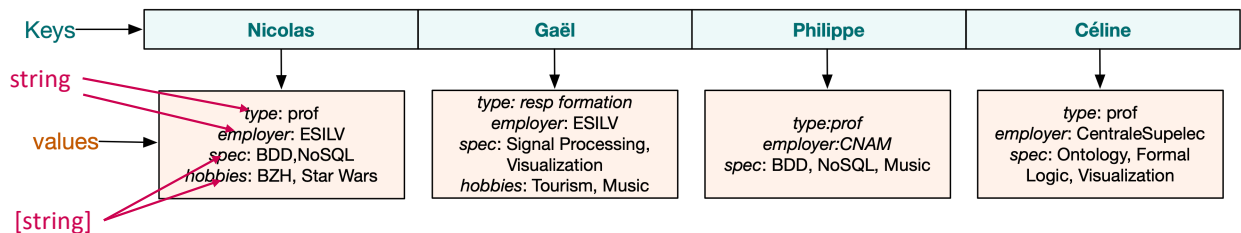


LIKE, SHARE, FOLLOW ! @esiv_paris | esilv.fr
Nicolas.travers@devinci.fr

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	CentraleSupélec	Ontology, Formal Logic, Visualization	



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

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	CentraleSupélec	Ontology, Formal Logic, Visualization	

id	status
Nicolas	prof
Gaël	resp formation
Philippe	prof
Céline	prof

id	employer
Nicolas	ESILV
Gaël	ESILV
Philippe	CNAM
Céline	CentraleSupélec

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 searches on products (ie. Ebay)
- Reporting

WIDE COLUMN-ORIENTED STORE: SOLUTIONS

Strong typing
Simple queries

APACHE
HBASE

Spark  SQL



Google
Big Query

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

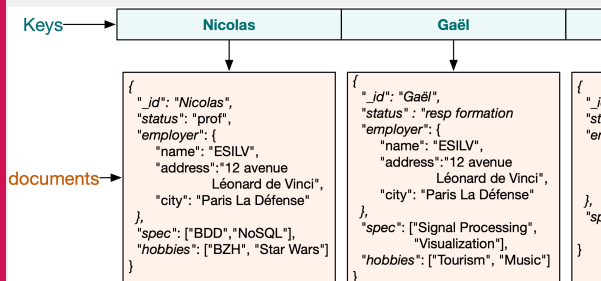
DOCUMENT-ORIENTED STORES

Based on the key-value store

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

HTTP API

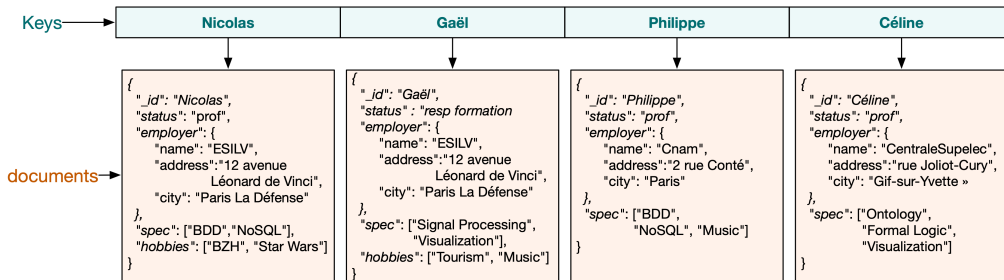
- More complex than CRUD
- Rich queries (database)



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

DOCUMENT-ORIENTED STORES: EXAMPLE

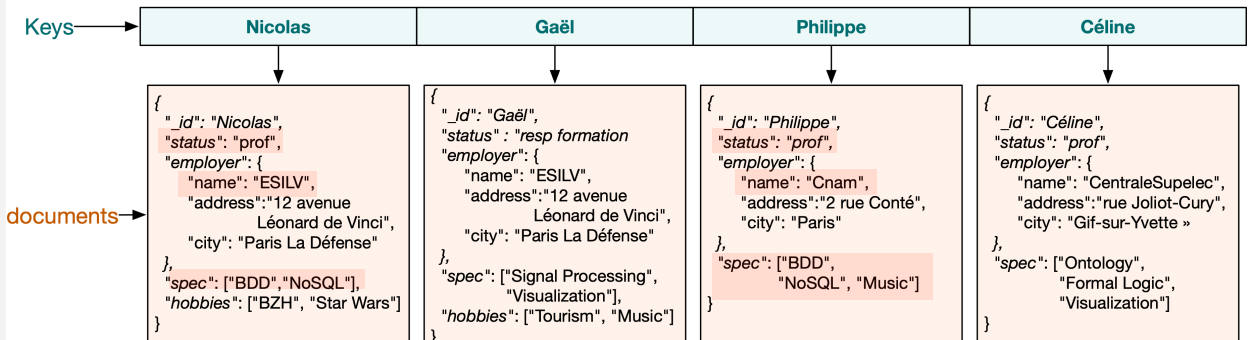
id	status	employer	spec	hobbies
Nicolas	prof	ESILV	BDD, NoSQL	BZH, Star Wars
Gaël	resp formation	ESILV	Signal Processing, Visualization	Tourism, Music
Philippe	prof	CNAM	BDD, NoSQL, Music	
Céline	prof	CentraleSupelec	Ontology, Formal Logic, Visualization	



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

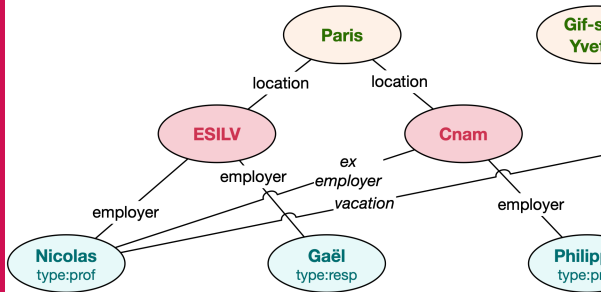


elasticsearch

GRAPH-ORIENTED STORES

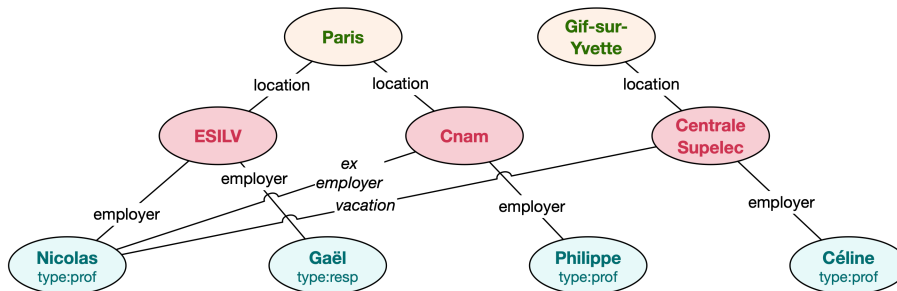
Storage: nodes, relations and properties

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



GRAPH-ORIENTED STORES: EXAMPLE

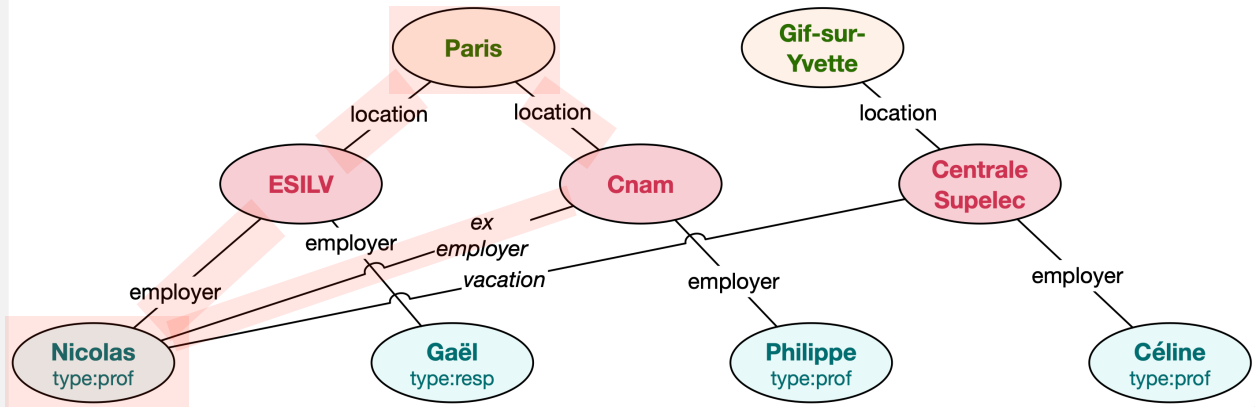
id	status	employer	spec	hobbies
Nicolas	prof	ESILV	BDD, NoSQL	BZH, Star Wars
Gaël	resp formation	ESILV	Signal Processing, Visualization	Tourism, Music
Philippe	prof	CNAM	BDD, NoSQL, Music	
Céline	prof	CentraleSupelec	Ontology, Formal Logic, Visualization	



GRAPH-ORIENTED STORES: QUERIES

Pattern queries

- Persons who had 2 employers at Paris



GRAPH-ORIENTED STORES: USE CASES

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

GRAPH-ORIENTED STORES: SOLUTIONS

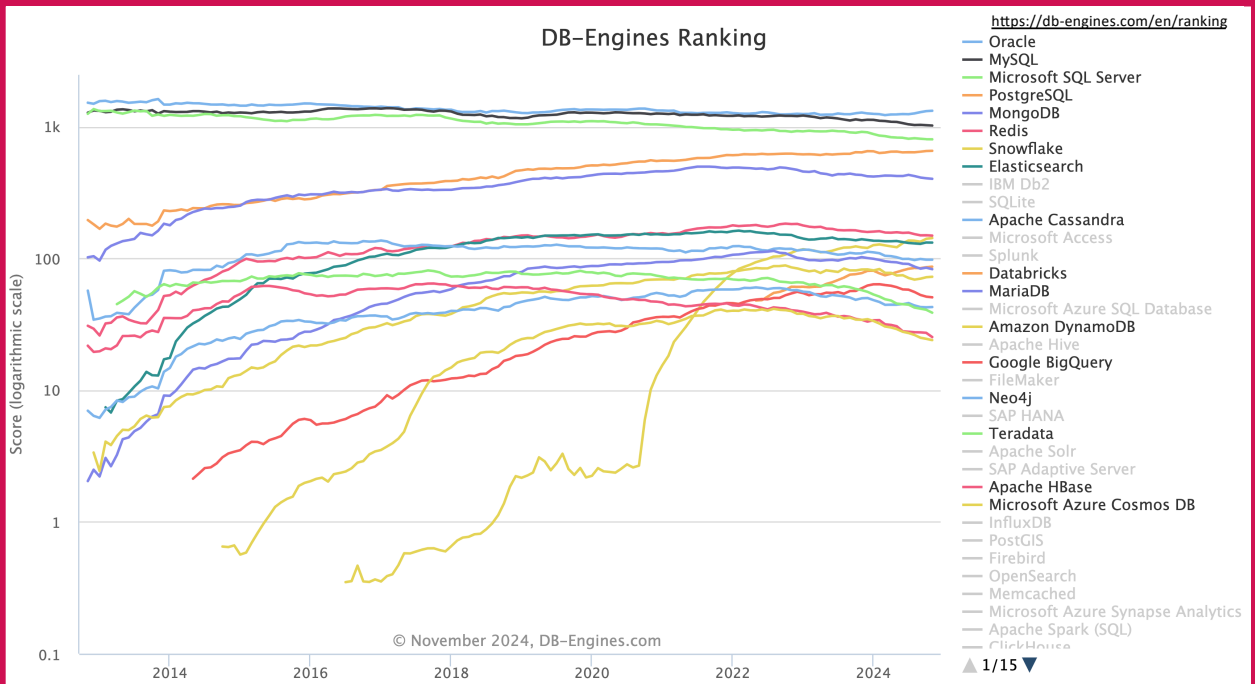
- Recommendation
- Network
- Knowledge Management



Azure Cosmos DB:



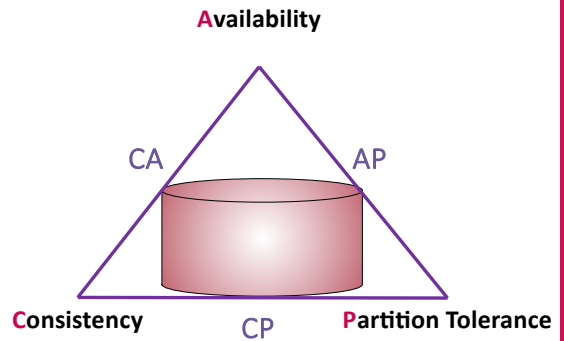
FlockDB



THE CAP THEOREM [BREWER 2000]

3 main properties for distributed management

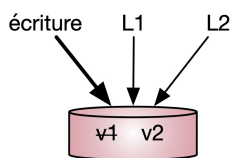
1. **Consistency:**
 - A data returns the proper value at any time (coherency)
2. **Availability:**
 - Even if a server is down, data is available
3. **Partition Tolerance:**
 - Even if the system is partitioned, a query must have an answer (unless for global failures)



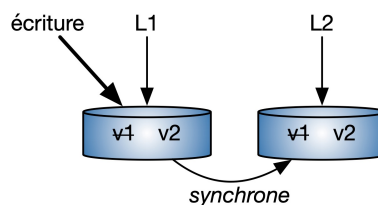
Theorem: A distributed, networked system can have only two of these three properties.

THE CAP THEOREM: ILLUSTRATION

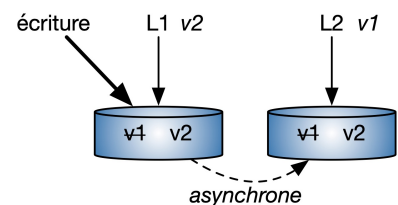
CA
Cohérence + Disponibilité



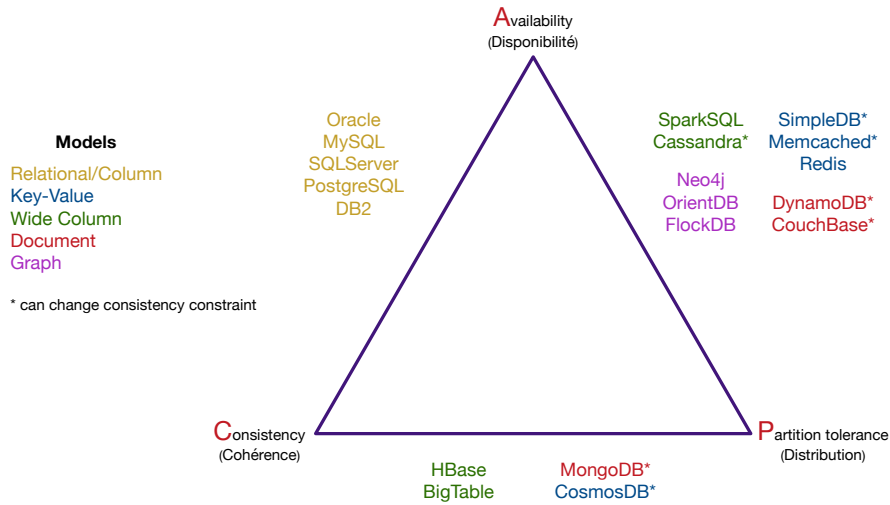
CP
Cohérence + Distribution



AP
Disponibilité + Distribution



THE CAP THEOREM: CAP TRIANGLE



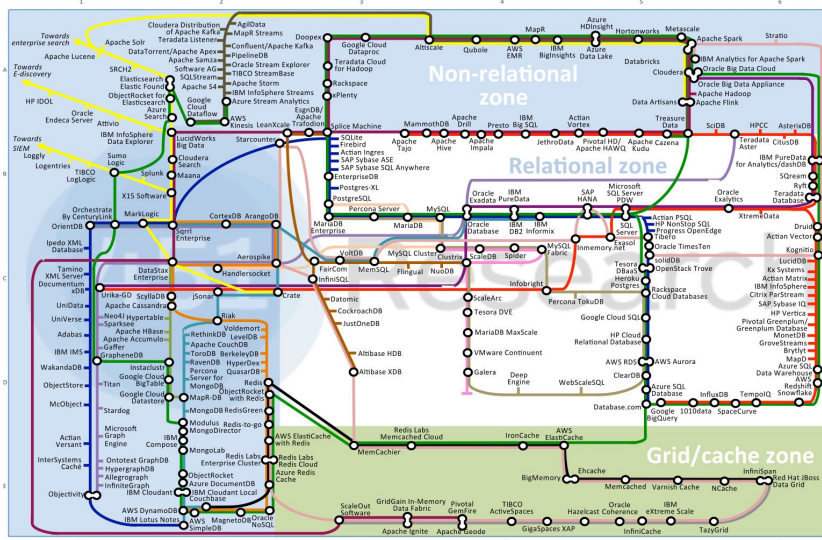
CAP VS PACELC [ABADI 2012]

Data synchronization

- **PAC**
 - Partitioning/Data movements
 - Between **A**vailability and **C**onsistency
- **ELC**
 - Normal state of the network
 - Between **L**atency and **C**onsistency
- **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



451 Research
Data Platforms Map
 January 2016

Key:

- General purpose
- Specialist analytic
- in-a-Service
- BigTables
- Graph
- Document
- Key value stores
- Key value direct access
- Hadoop
- MySQL ecosystem
- Advanced clustering/sharding
- New SQL databases
- Data caching
- Data grid
- Search
- Appliances
- In-memory
- Stream processing

<https://451research.com/state-of-the-database-landscape>

© 2016 by 451 Research LLC. All rights reserved.

SQL vs. NoSQL: Cheatsheet for AWS, Azure, and Google Cloud

	AWS	Azure	Google Cloud	Cloud Agnostic
ACID Transactions (OLTP)	RDS, Aurora	Azure SQL Database	Cloud SQL, Cloud Spanner	SQL Server, Oracle, DB2, MySQL, PostgreSQL
Analytics (OLAP)	RedShift	Azure Synapse	BigQuery	Snowflake, ClickHouse, Druid, Pinot, Databricks
Dictionary	DynamoDB	Cosmos DB	BigTable	Redis, ScyllaDB, Ignite
Cache	ElastiCache	Azure Cache for Redis	MemoryStore	Redis, Memcached, Hazelcast, Ignite
2-D Key-Value	Keyspaces	Cosmos DB	BigTable	HBase, Cassandra, ScyllaDB
Time Series	TimeStream	Cosmos DB	BigTable, BigQuery	TimescaleDB, OpenTSDB, InfluxDB, ScyllaDB
Audit Trail	Quantum Ledger Database (QLDB)	Azure SQL Database Ledger		Hyperledger Fabric
Immutable Ledger	Keyspaces	Cosmos DB	BigTable, BigQuery	Solr, PostGIS, MongoDB (GeoJSON)
Location & Geo-entities	Keyspaces	Cosmos DB	BigTable, BigQuery	Solr, PostGIS, MongoDB (GeoJSON)
Entity-Relationships	Neptune	Cosmos DB	JanusGraph + BigTable	OrientDB, Neo4J, Graph
Nested Objects (XML, JSON)	Document DB	Cosmos DB	Firestore	MongoDB, Couchbase, Solr
Document	OpenSearch, CloudSearch	Cognitive Search	Search APIs on Datastores	ElasticSearch, Solr, Elasticsearch
Full Text Search (Rich) Text	S3	Blob Storage	Cloud Storage	HDFS, MinIO
Blob				

DATABASE CHOICE

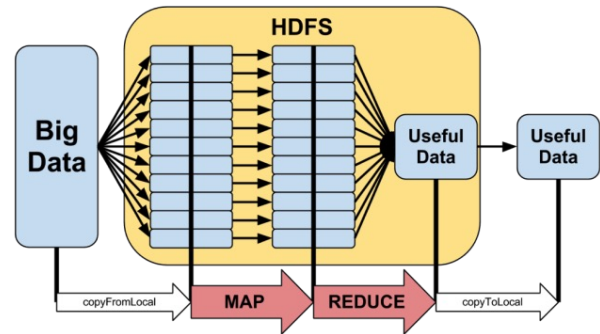
Credits - Satish Chandra Gupta

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**

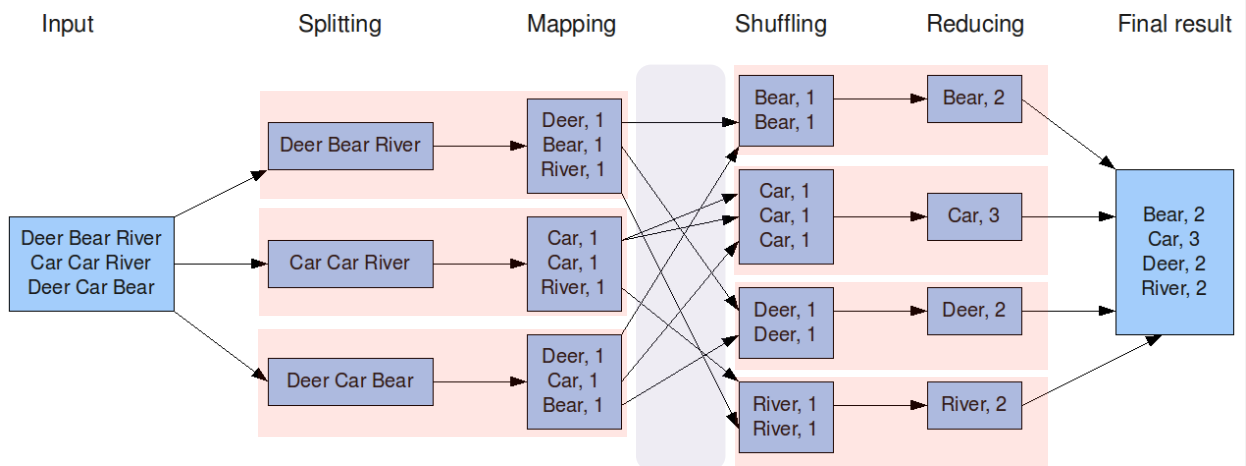


Scalability, Fault Tolerance

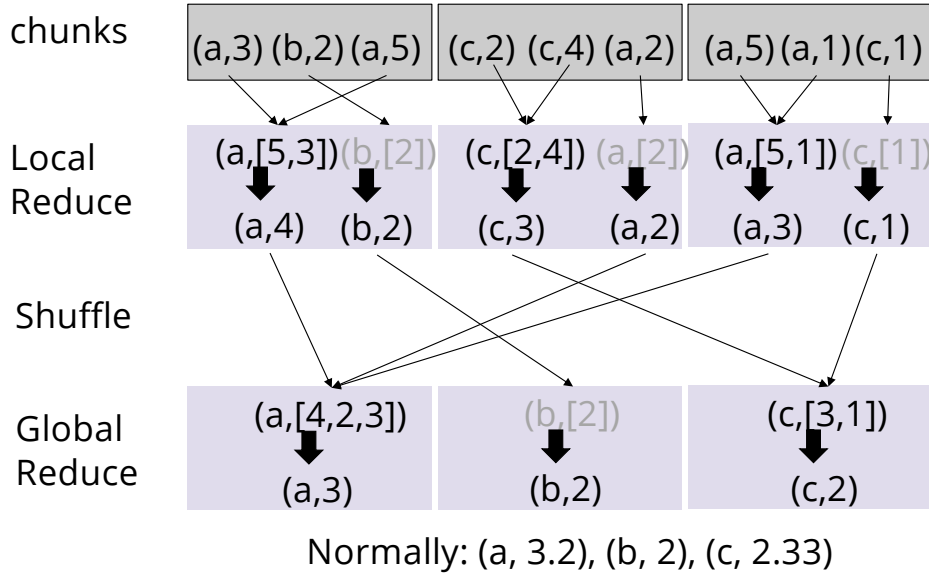
- Send map&reduce to the cluster (jobs)
- In case of failure of 1 job, restart the part of the job on another server

MAPREDUCE: WORD COUNT EXECUTION

The overall MapReduce word count process



REDUCE: AVERAGE FUNCTION (1/2)



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

MAPREDUCE: PROPERTIES

Programming language

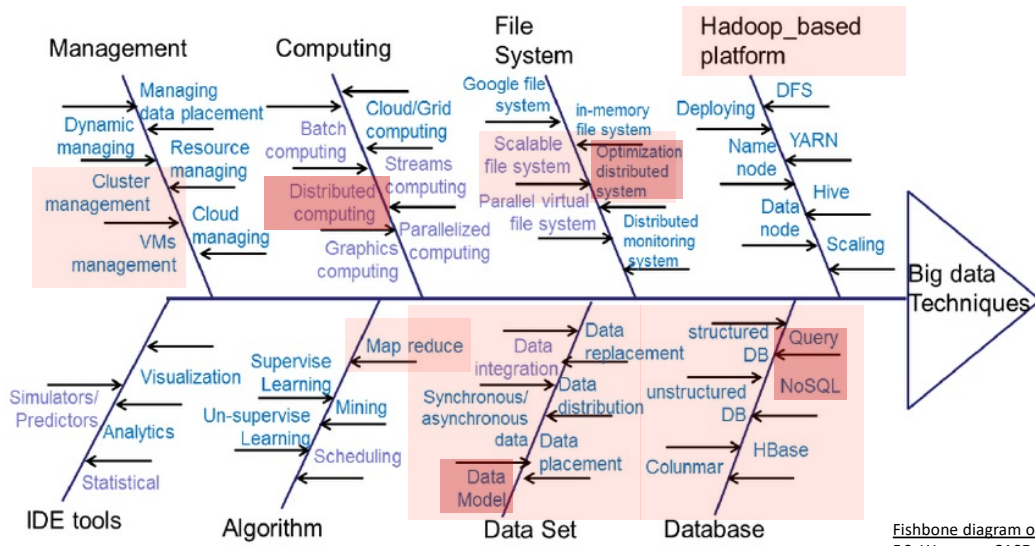
- Highly distributable
- Fault tolerant
- Simple to express
- **NoSQL** -> on-top of MapReduce (operations are developed in MR)

Drawbacks

- No join queries
- Many networks communications (shuffle)

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

BIG DATA OVERVIEW – COURSE'S FOCUS



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