Research Project Proposal: Towards a unifying model for data-intensive applications

Nicolò Felicioni nicolo.felicioni@mail.polimi.it Computer Science and Engineering





Data-intensive applications

What is a «data-intensive» application?

We are talking about:

- Software applications
- Big data

The Big Data era

Big data means (at least) three problems:

- 1. Big Volume
- 2. Big Velocity
- 3. Big Variety

THE 3Vs OF BIG DATA

VOLUME

- Amount of data generated
- Online & offline transactions
- In kilobytes or terabytes
- Saved in records, tables, files



VELOCITY

- Speed of generating data
- Generated in real-time
- Online and offline data
- In Streams, batch or bits

VARIETY

- Structured & unstructured
- Online images & videos
- Human generated texts
- Machine generated readings

Ð

Data-intensive vs. Compute-intensive

Data-intensive application: data (the quantity, the speed at wh primary challenge

Compute-intensive application: CPU is the bottleneck

data (the quantity, the speed at which it is changing, the variety) is the

Now a step back into the state of the art

Two main areas:

- Database research area
- Distributed systems research area

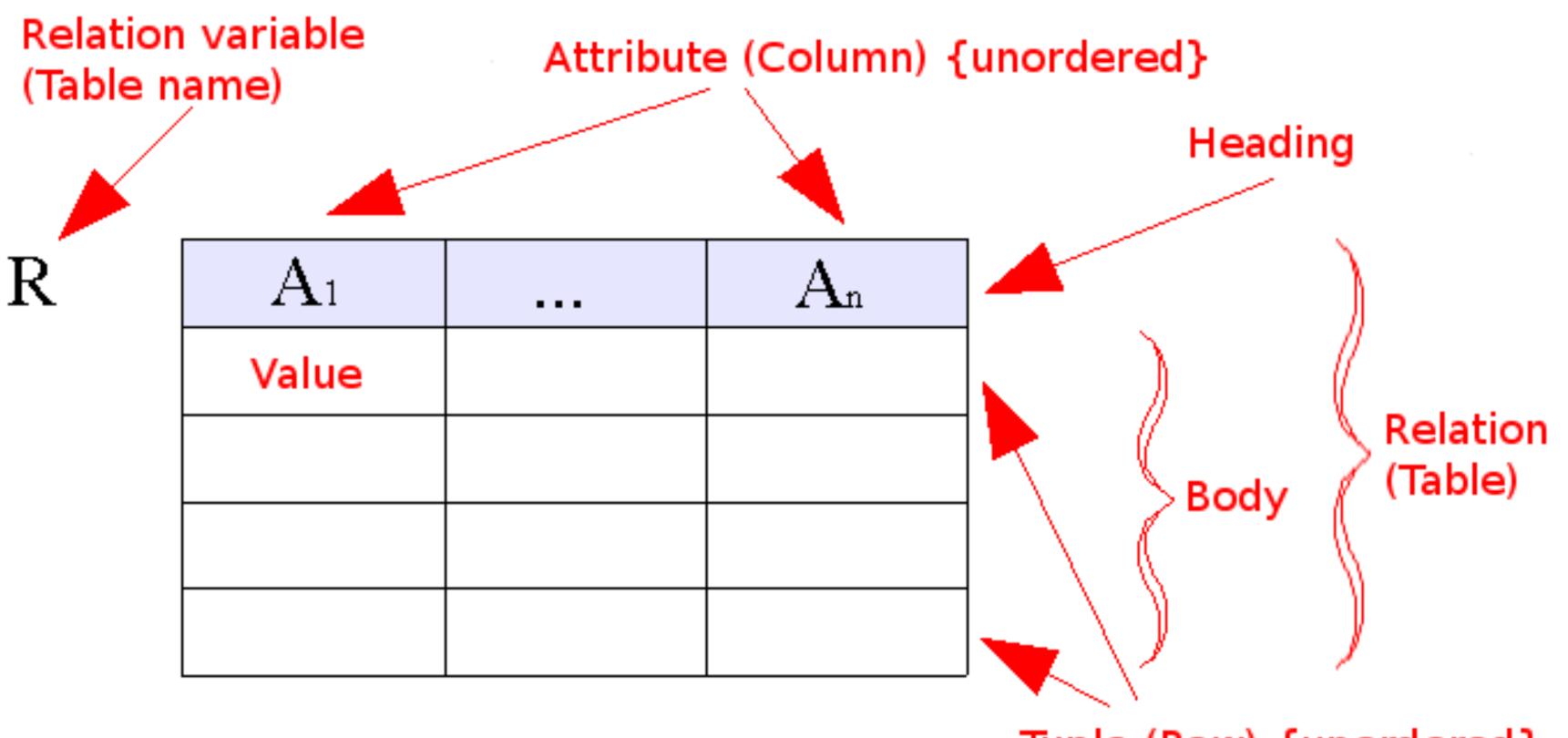
A tale of two worlds

Database basics

- Collection of data
- (DBMS)
- The first data model was the relational model

• Software used to manage databases is called Database Management System

Relational model



Tuple (Row) {unordered}

Transactions

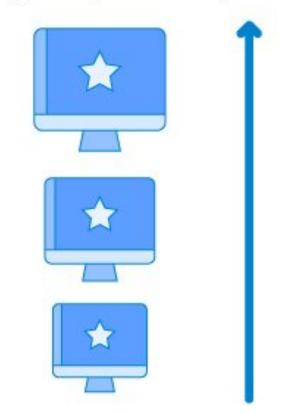
- Classical DBMSs usually support transactions
- A transaction is a unit of work that must be Atomic, Consistent, Isolated and Durable (ACID)
- On-line Transaction Processing (OLTP) is a scenario where a database is used mainly for processing multiple transactions
- The transaction management can be a bottleneck when implemented in data-intensive systems

Issues with early databases

• Classical solutions (Oracle, MySQL) were not good at "horizontal" scaling

VERTICAL SCALING

Increase size of instance (RAM, CPU etc.)



• A new type of systems called NoSQL started to gain relevance in the 2000s

Image from:https://www.redswitches.com/hs-fs/hubfs/scaling-image.jpg?width=1580&name=scaling-image.jpg

HORIZONTAL SCALING

(Add more instances)





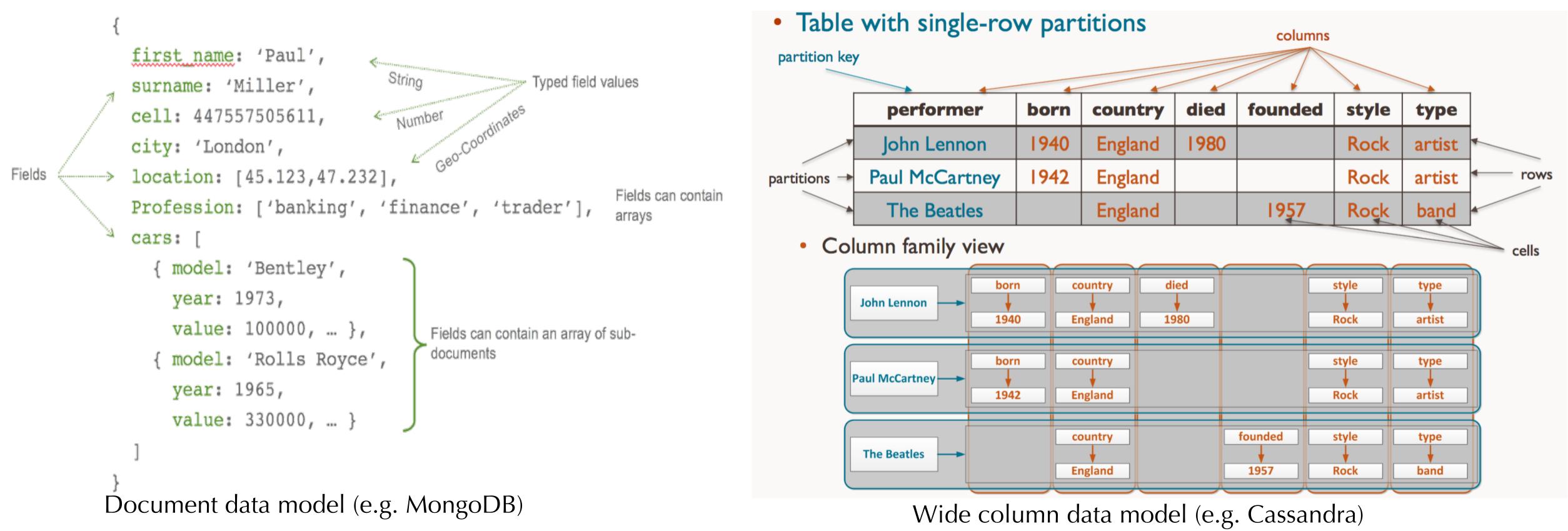
NoSQL started for data-intensive needs – Volume, Variety Usually a NoSQL database is:

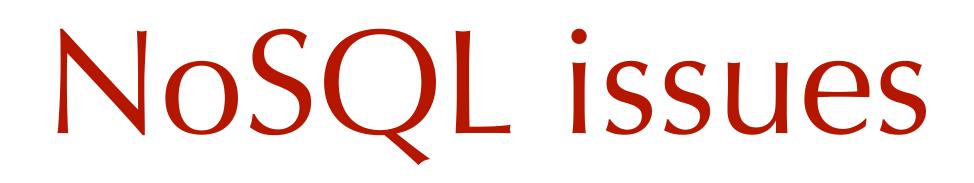
non-relational

- distributed
- open-source
- horizontally scalable

NoSQL data models

NoSQL is an inherently heterogeneous category





- Though they have a big flaw: lack of support for full ACID transactions
- And OLTP market is still relevant

• NoSQL systems are valuable tools, especially for data-intensive requirements

- The solution for scalable OLTP scenarios: NewSQL
- They try to make scalable as much as possible the traditional relational systems, while preserving all their guarantees



• Different approaches were adopted to implement transactions with strong consistency and isolation with sufficient performance and availability

- Synchronization based on specialized hardware like atomic clocks, adopted by Google Spanner
- Limit transaction expressivity, adopted by Calvin
- Using information on replication provided explicitly by the user the to optimize transactions in distributed settings, adopted by VoltDB

NewSQL approaches

Two main areas:

• Database research area

• Distributed systems research area

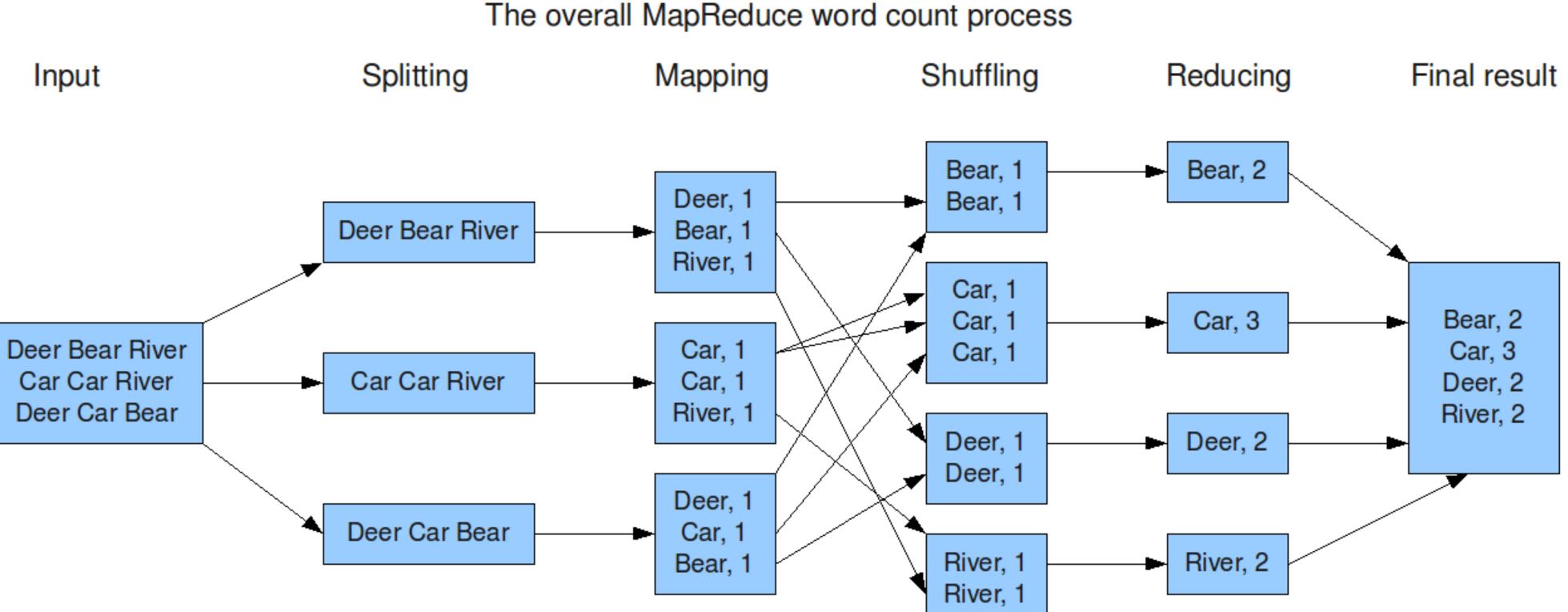
A tale of two worlds

MapReduce

- In Distributed Systems research, systems explicitly designed for distributed processing in large-scale compute infrastructures started to gain popularity
- These systems trace their roots to Google's programming model called MapReduce (2004)

MR fundamentals

- The computation is split into two phases, Map and Reduce
- Map processes individual elements For each of them outputs one or more <key, value> pairs
- **Reduce** processes all the values with the same key and outputs a value
- The runtime system controls scheduling, load balancing, communication, fault tolerance



MR word count example

Beyond MapReduce

In the last decade, many systems extended and improved the MapReduce abstraction in many ways

- From two processing steps to arbitrary acyclic graphs of transformations
- From batch processing to stream processing
- From disk to main-memory or hybrid approaches

Examples:

- Apache Spark for batch processing
- Apache Flink for stream processing

Batch processing - Spark

- Similar to MapReduce
 - O Instead of only two stages (map and reduce) ...
 - O ... arbitrary number of stages
- Intermediate results can be cached in main memory if they are reused multiple times
- the data

• Scheduling of tasks (stages) ensures that the computation takes place close to

Stream processing - Flink

- A job is not split into stages that are scheduled
- Instead, all the operators are instantiated as soon as the job is submitted
 - O They communicate using TCP channels
 - O An operator can start processing as soon as it has some data available from the previous ones
 - Pipeline architecture where multiple operators are simultaneously running

Data-intensive issues

- The presented data systems –relational DB, NoSQL, NewSQL, MR, batch/stream processing– offer solutions to solve <u>specific</u> data processing and management tasks
- But often requirements of a data-intensive application can be heterogeneous
- Therefore they cannot be satisfied by any of these systems <u>alone</u>

- Developers in practice build complex architectures that combine multiple systems
- They implement application logic in order to orchestrate their interaction

Current approach

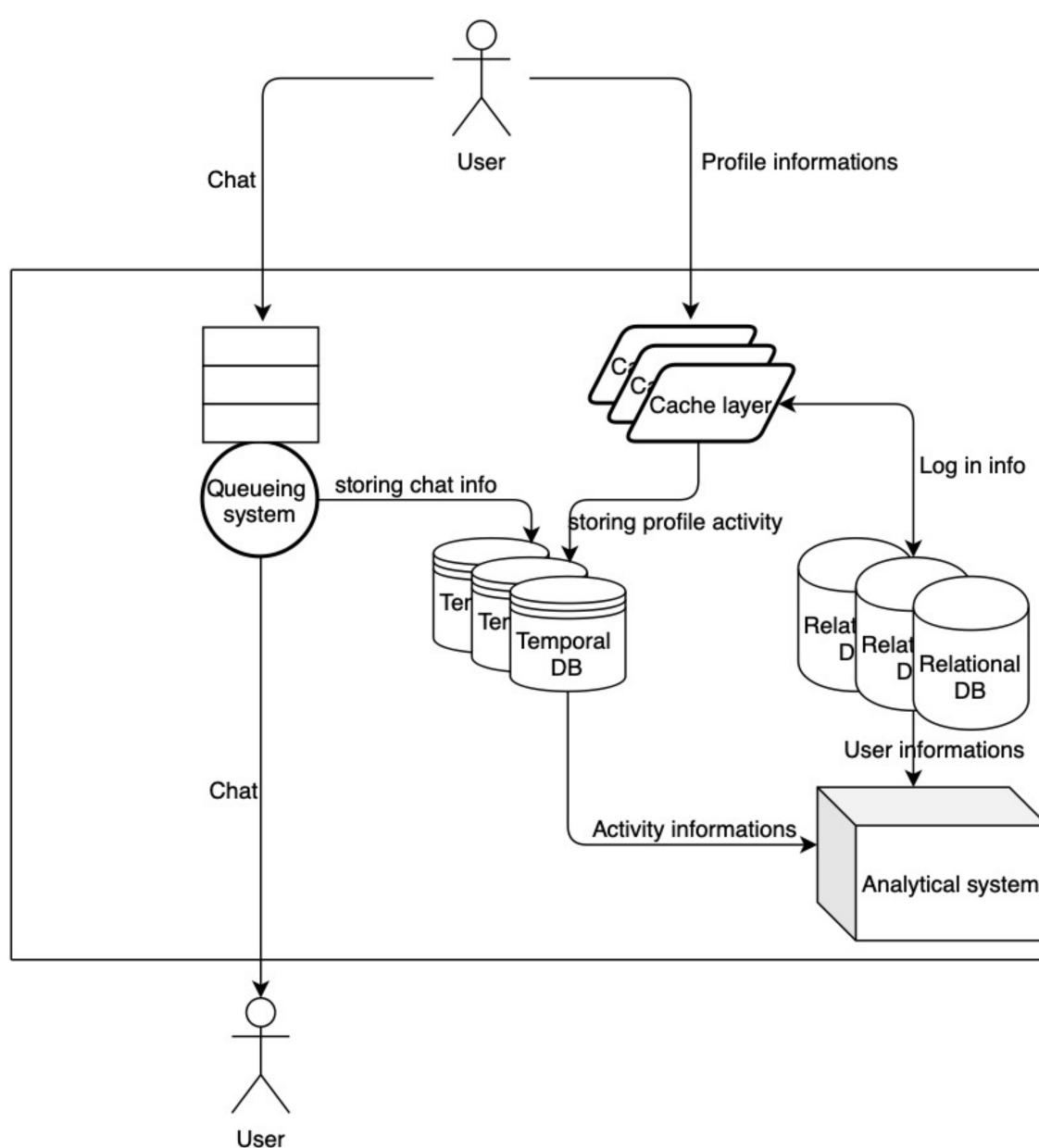
Current problems

- In doing so, they lose the benefits provided by the systems in terms of guarantees on the data and transparent deployment and communication
- Also, integrating data systems together necessitates a deep understanding of:

o Semantics

- o Workload assumptions
- o Performance characteristics
- Deployment strategies Ο
- o Configuration opportunities

An online collaboration tool example



Developers need to :

- configure individual subsystems
- manually integrate the subsystems
- implement the mechanisms that ensure correctness criteria (profile information is consistent across replicas, temporal database and the queuing system have consistent order of messages, ...)
- Take care of performance concerns

A unifying model

- The goal of the research is finding a formal model that defines high-level notions and structures
- The purpose is twofold:
 - 1. the various data-intensive systems usually present intersections among them, therefore a unifying model can be useful to better understand the semantics of the converging concepts of different systems
 - 2. this modeling framework can be a first fundamental step in the direction of a <u>change of paradigm</u>, that leads to a new approach for designing data-intensive application

A unifying model

• In this way, developers no more have to deal with trying to put different and independently developed systems together in a sort of "software collage", where the formal guarantees provided by the single systems could be lost.

A unifying model

Conceptual model

U	oncepte	ai mouei															1		
Time	Event time	Proceeding time	Yes	Yes Yes Operator varial	Yes Ves Yes Operator variables	Ven Ven Ven Transformed data -	Yes Yes Yes	ar lables	Yes No Yes Operator variables		Clients interactions	 Code deployment Clients requests Send and receive 	 Code deployment Send and receive data items 	- Code d - Send a data ite	leployment rd receive	 Clients requests Send and receive data items 	 Code deployment Client requests Send and receive 	 Client request Code deployment Send and receive 	
		Joppat data	Data items	Data itema Data itema		Data items Data items			Data items			data items	0.000 100100			CARGA DECIDA	data items	data items	
Data set	Inematable	Transformed data			items Transformed data item	to Transformed data items	s Transformed data items		Transformed data items		Ingestion time clock	Yes	Yes	Yes		Yes	Yes	No	
		Output data	 Transformed data Operators variable 		ta itema Transformed data item	a Transformed data item	 Transform Operators 	sed data items variables	 Transformed data items Operators variables 		Operational mode	Pipelined Scheduled		Schedul	ed	Scheduled	Pipelined	Scheduled	
	Data topology	Partitioned data		Yes	Yes	Yes	Yes		Yes				V			hi.	N general	N-	
		Replicated data		Yes	Yes	Yes	Yes		Yes		Leverage data locality	740	105	145		760	760	Tes	
		Data schema	Date schema - Custom schema - Custom schema Table scheme - Table scheme - Table scheme DS statement - Ciant request Code deployment			1									ent time	Run-time	 Deployment time Run-time (KSQL) 	 Run-time Deployment time 	
		D5 statement			e C														
		Administrative action Input action		 Via code deploy No 								1				Imperative	 Imperative Declarative (KSQL) 	Declarative	
Det	Dete action		information action No. No.		N										d + declarative	Imperative	Functional	Imperative + declarative	
		Output action	Output action Via client request No.		N												- Client request	- Client request	
Continuous - D		Input action		Yes	1					~				3	loyment	Client request	- Code deployment	- Code deployment	
	Data action D	of a framification action	Output action Yes Yes Y			$\neg < =$											- Run-time	- Run-time	
		Transaction	- No	No	P		~									Run-time	- Deployment time	- Deployment time	
			- Yes (Streaming Le				\sim										L. reductioners must	. reductioners must	
		Computational topology	Seator.	Dynamic	1s														
Ar	chitectu	iral model	l														Guara	ntees model	
		- Internal	- In		- Inter										quest	0	ontinuous d	ata actions	
Re	юштсе тапа	0.079	- External - Extern												-				
		- 1.4.001100	- 1.4		- 10400											No			
1	lorker manaj	per Yes	Yes		Yes											Not	supported		
		_													_				
	Execution 1	dot Yes	Yes	Yes Yes		Yes	Yes		Yes			Durability	erability N / A				Recoverable — same state		
		_									Replication consistency N / A					Seco	Securitial		
17000	ioned resour	Deploym	ent time Dep	Deployment time Deployment tim		Run-time	Deployn	nent time	Run-time		Reprication consistency		N/ N			Sequential			
	modificat	101										-end order Sequential			FIF	FIFO by key			
		ion Computa	tion Dat		Deta	Data Computation		ation	Data										
		_			1.4158		Computation					nd delivery	NZA				least once		
0	sde deploym	ent Required	Reg	aired	Required	Not possible	Required		Optional			End-to-end delivery				- Ex	- Exactly once		
																-			

Operational model

1. Scope definition

- 1. Scope definition
- 2. Systems identification and classification





- 1. Scope definition
- 2. Systems identification and classification
- 3. Preliminary study of the tools

- 1. Scope definition
- 2. Systems identification and classification
- 3. Preliminary study of the tools
- 4. First model

		familie time 1		Yes	Yes	Yes	Yes	Yan		- Code deployment					- Code deployment	- Client request
Tow .	Eneret time -	Appendix law V		Yest	12m	Yes	Yes	No		- Clients requests	- Code deployment		eployment	- Clients requests	- Chand monotonic	- Code deployment
		Proceeding closer W	Presenting time Yes		Ves	No.	Ves	Yes	Clients interactions	- Send and receive	- Send and receive	- Send a	sd receive	- Send and receive	- Send and receive	- Send and receive
	Mutable		perator variables	Operator variables	Operator variables	Transformed data -	Operator variables	Operator variables		- Send and receive data items	data itema	data ite	ma	data itema	data items	data itema
Deficient		Input data D		Data items	Chala Berra	Data items	Chatta menera	Data items				_			cara recira	claria recerca
	International -		analormad data terra l'analormad data terra	Transferred data ite	na Transformed data News	Transformed data item	Transformed data in	term Transformed data items feets - Transformed data items	Ingestion time clock	Yes	Yes	Yes		Yes	Yes	No
		Output data	Operators variables Transformed data	Transformed data to	ru Transformed data items	to Transformed data item	Operators variables		Operational mode	Produced	Scheduled	Schedul	ad be	Scheduled	Pipelined	Scheduled
	Date topology -	Partitional data 1		Yes	Ves	Yes	Yes	Tes							- general	
		Replicated data Y		Yes	Ten	Yes	Yan	Yes	Leverage data locality	No	Yes	Yes		No	No	Yes
		- Date scheme	able afterna able afterna	-Custom scheme Table scheme	-								ent time	Run-time	 Deployment time Run-time (KSQL) 	- Ran-time - Deployment time
		DS statement	Samt request	Code deployment	1					_					and the second se	- referiren one
		Administrative action V Input action N		Via code deployeesee							-			Imperative	- Imperative - Declarative (KSQL)	Declarative
	Carto antica	Transformation action N			-6			<u> </u>		-			d + declarative	Imperative	Functional	Imperative + declarativ
		Output action V		No.	-											
-		Input action Vi		Yes	-	_							loyment	Client request	- Client request	- Client request
	- Data action	Data transforming action Y											-		 Code deployment 	 Code deployment
		Output action 'V		Yes			~				1011 10			Run-time	- Run-time	- Run-time
		Transaction	to (millionizering Ledger)	No	1 M	1						-		And the second	- Deployment time	- Deployment time
		Computational topology (b)		Dynamic						5		5				
A	rchitect	ural model													Guara	ntees model
		- Internal	- Internet		i and								quest	1.0	antinuous d	lata actions
Resource manager		Lines and Lines		Internal - Inter									dacar	~	Continuous data action No	
		- External	External External		xter									No		
_		. 1.4.0011.001			and the second se											
_	Worker man		Yes	Ye										Not	supported	
_	Worker man	ger Yes		Ye		V-	Vor	Vor		Durability	N / A				supported	same state
_		ger Yes	Yes			Yes	Yes	Yes		Durability					supported overable —	same state
	Worker man	ager Yes slot Yes	Yes	Ye	1				Replication of					Rec		same state
	Worker man Execution	eger Yes slot Yes rors Darchard	Yes	Ye		Yes Run-time		Yes ime Run-time	Replication of	onsistency	N / A]	Rec	overable — uential	same state
	Worker man Execution isioned resou modifica	eger Yes slot Yes rces Deploymen	Yes t time Deploy	Ye Ye ment time De	s ployment time	Run-time	Deployment t	ime Run-time	Replication of	onsistency				Rec Seq FIF	overable — uential O by key	same state
	Worker man Execution isioned resou modifica	eger Yes slot Yes rors Darchard	Yes t time Deploy	Ye	s ployment time				Replication o End-to	onsistency	N / A Sequential			Rec Seq FIP	overable — uential	same state

- 1. Scope definition
- 2. Systems identification and classification
- 3. Preliminary study of the tools
- 4. First model
- 5. Experiments and consolidated model (iterative task)

- 1. Scope definition
- 2. Systems identification and classification
- 3. Preliminary study of the tools
- 4. First model
- 5. Experiments and consolidated model (iterative task)
- 6. Writing

	2019		2020								
Task Name	Nov	Dec	Jan	Feb	Mar	Apr	May	June	July	Aug	Sept
Scope definition				_	•						
Systems' identification and classification						_					
Preliminary study of the tools											
First model											
Experiments, consolidated model and iterations											
Writing											

