Contribution to the Analysis of the Design-Space of a Distributed Transformation Engine

Jolan PHILIPPE

PhD Defense, speciality: Computer Science

Referees:	Jesús SÁNCHEZ CUADRADO	Associate professor, Universidad de Murcia, Spain
	Matthias TICHY	Professor, Ulm University, Germany
Examiners:	Thomas LEDOUX	Professor, IMT Atlantique, France
	Leen LAMBERS	Professor, Brandenburg University of Technology, Germany
	Antonio VALLECILLO	Professor, University of Málaga, Spain
Ph.D. director:	Gerson SUNYE	Associate professor, University of Nantes, France
Ph.D. advisors:	Hélène COULLON	Associate professor, Institut Mines-Telecom Atlantique, France
	Massimo TISI	Associate professor, Institut Mines-Telecom Atlantique, France

lowcomote LS2N

19th December 2022

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

naomod

Outline

DOCTORAT BRETAGNE LOIRE MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS-DE-LA-LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE Nº 601
Mathématiques et Sciences et Technologies
de l'Information et de la Communication
Spécialité : *Informatique*

Par
Jolan PHILIPPE

Contribution to the Analysis of the Design-Space of a Distributed Transformation Engine

Thèse présentée et soutenue à Nantes, le tbd
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes
Thèse Nº : tbd

Rapporteurs avant soutenance :

Jesus SANCHEZ CUADRADO	Associate professos, Universidad de Murcia, Spain
Matthias TICHY	Professor, Brandenburg University of Technology, Germany

Composition du Jury :

Président :	Thomas LEDOUX	Professor, IMT Atlantique, France
Examinateurs :	Leen LAMBERS	Professor, University of Ulm, Germany
	Antonio VALLECILLO	Professor, University of Málaga, Spain
Dir. de thèse :	Gerson SUNYE	Associate professor, University of Nantes (France)
Co-dir. de thèse :	Massimo TISI	Associate Professor, Institut Mines-Telecom Atlantique (France)
	Hélène COULLON	Associate Professor, Institut Mines-Telecom Atlantique (France)

1 CONTEXT & MOTIVATION

2 CONTRIBUTIONS

2.1 SPARKTE: A DISTRIBUTED TRANSFORMATION ENGINE

2.2 DISTRIBUTED QUERY EVALUATION STRATEGIES

2.3 FEATURE ANALYSIS

3 CONCLUSION

1 CONTEXT & PROBLEMATIC

Context of the thesis

lowcomote

Lowcomote is a H2020-ITN project aiming at training 15 PhD students, and build a low-code development platforms based on

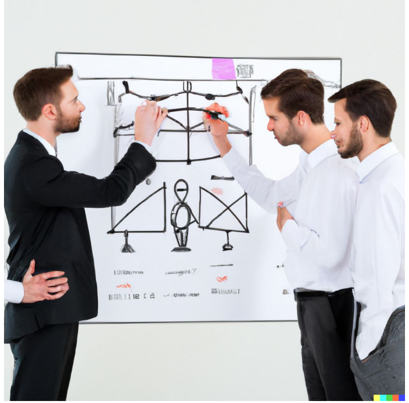
- **Model-Driven Engineering**
- **Cloud Computing**
- **Machine Learning**

IMT Atlantique, University of York, UAM, Università degli Studi dell'Aquila, JKU Johannes Kepler Universität Linz, BT, intecs Solutions, UGROUND, CLMS, IncQuery Labs, Sparx Systems, METADEV, The Open Group, aws

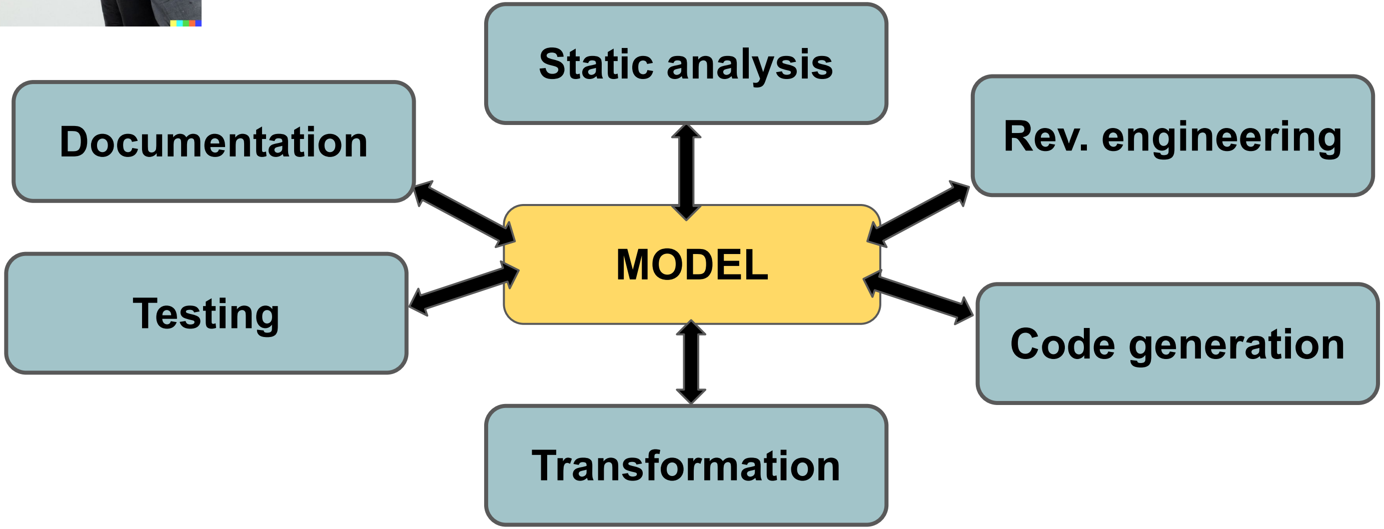
Marie Skłodowska-Curie Actions

WP5: Scalable Low-Code Artefact Management

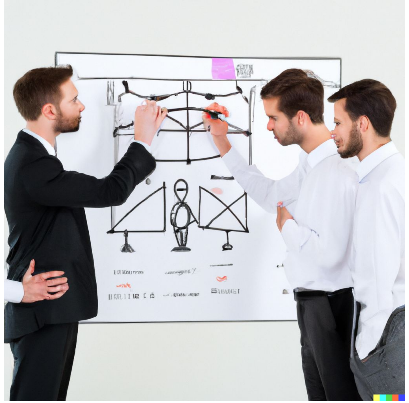
Model-Driven Engineering



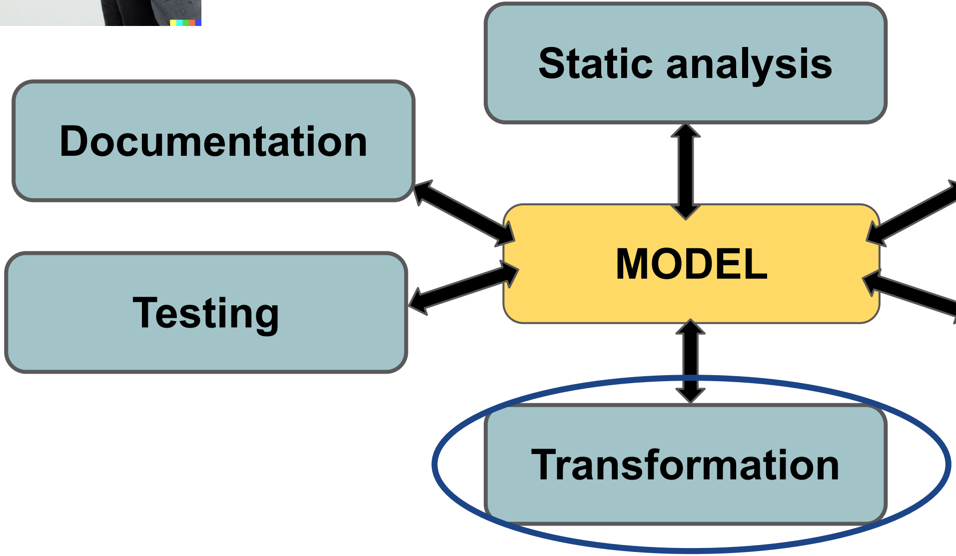
- **Software engineering** approach
- Models as the **central artifact** to represent systems



Model-Driven Engineering



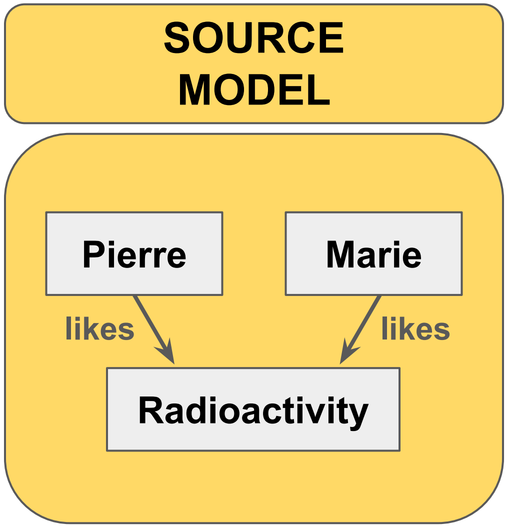
- **Software engineering** approach
- Models as the **central artifact** to represent systems

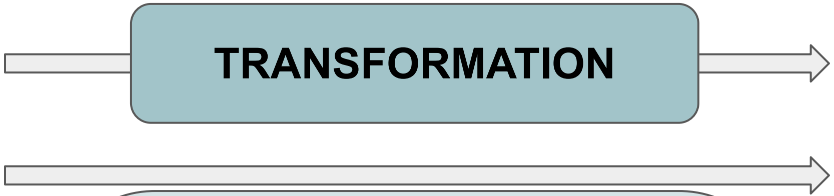


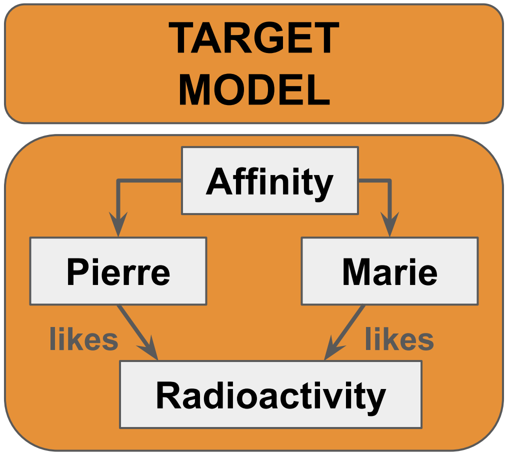
Model Transformation

Model Transformation

Model Transformation

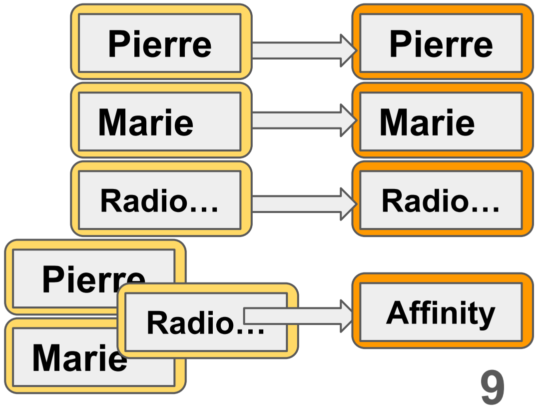






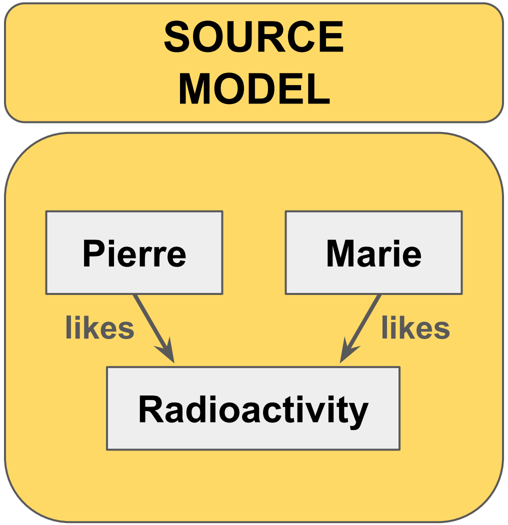
rule copy (e: Element)
output:
new Element (content ← e.content)

rule affinity (e_1: Element,
e_2: Element)
matching:
$(e_1.likes) \cap (e_2.likes) \neq \emptyset$
output:
new Affinity (from ← e_1, to ← e_2)



Many transformation languages: ATL, ETL, QVT, Henshin, Viatra, …

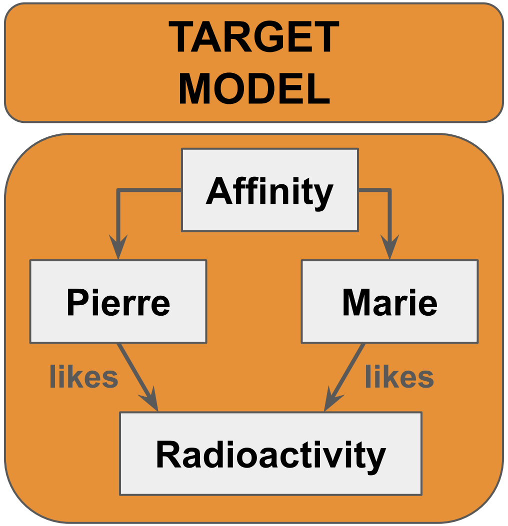
Model Transformation



TRANSFORMATION

rule copy (e: Element)
output:
new Element (content ← e.content)

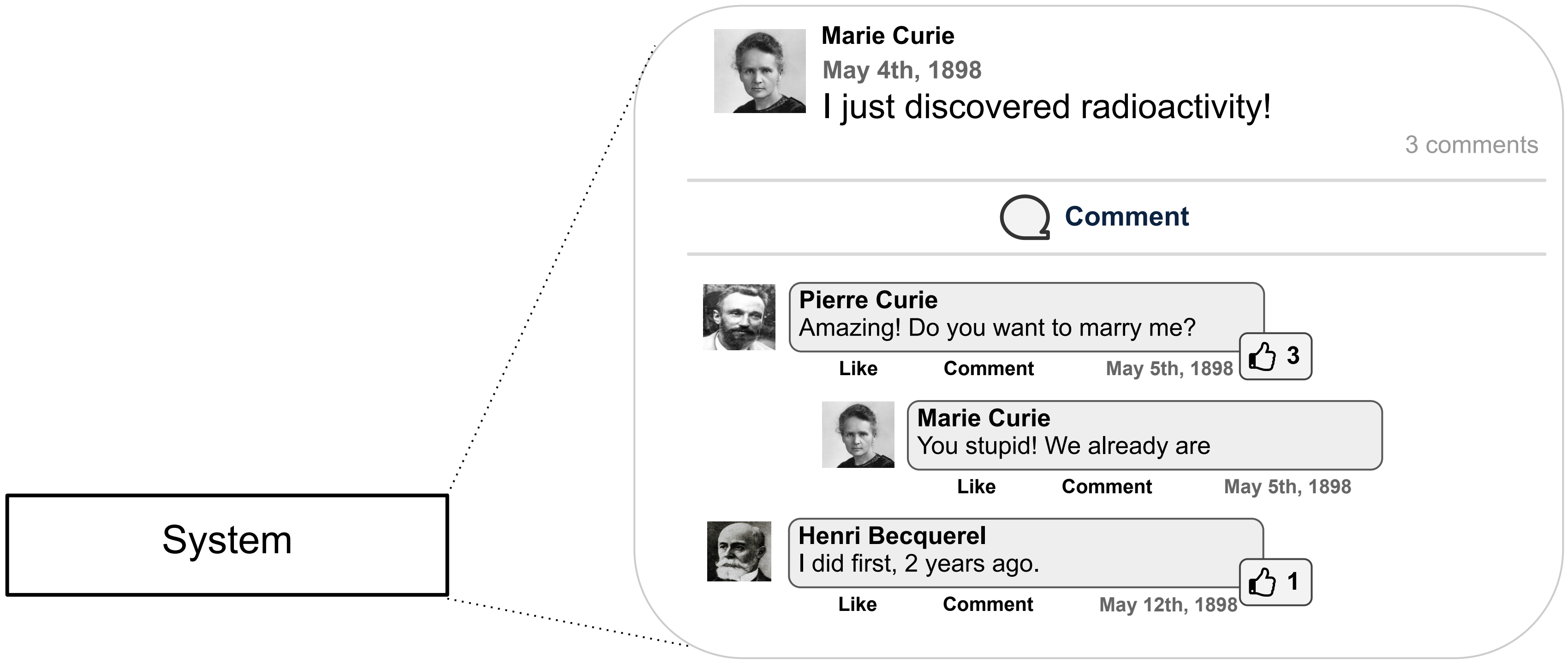
rule affinity (e_1: Element,
e_2: Element)
matching:
$(e_1.likes) \cap (e_2.likes) \neq \emptyset$
output:
new Affinity (from ← e_1, to ← e_2)



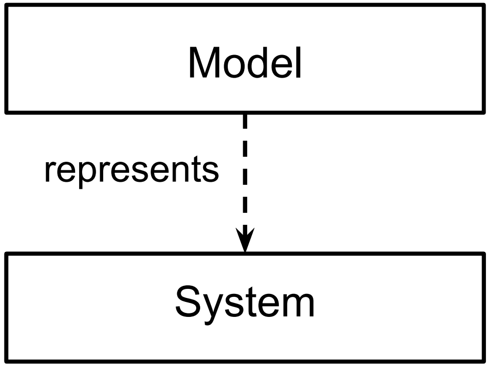
The **expression** e_i.likes can be expressed as a **query**

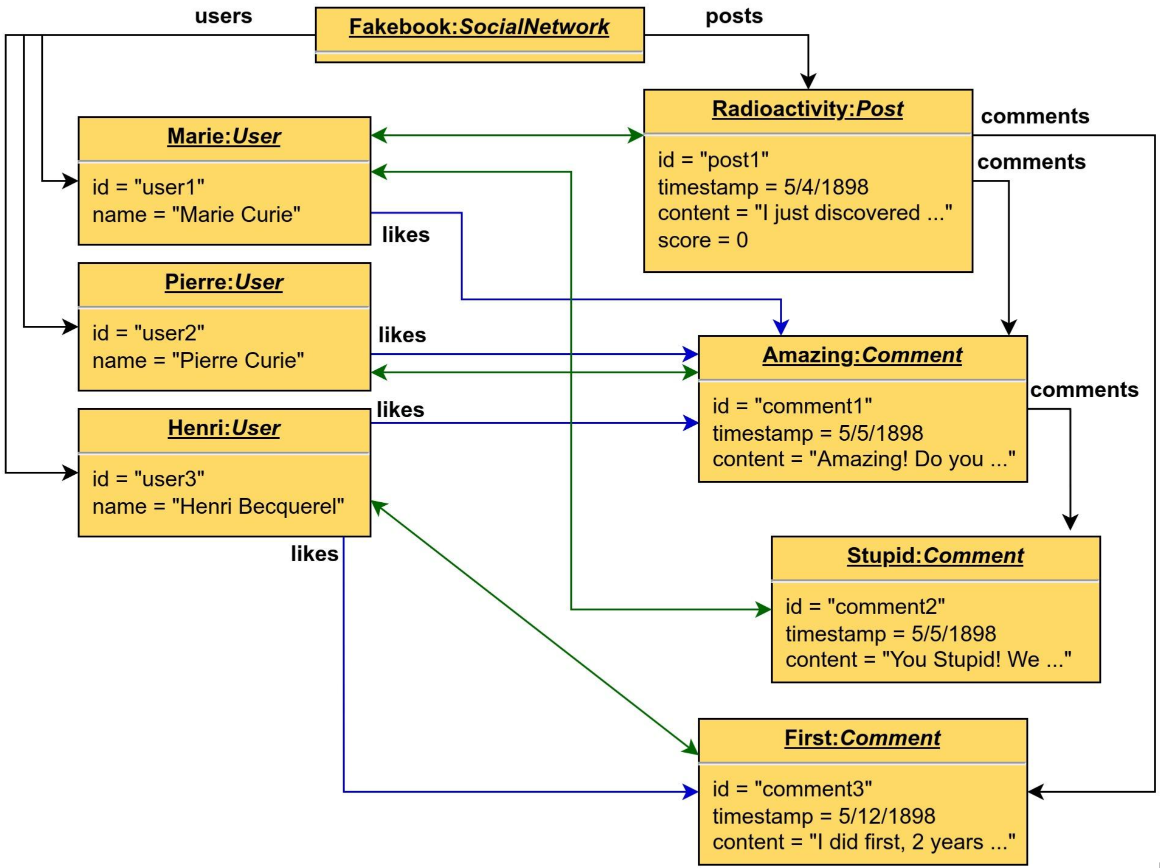
Many transformation languages: ATL, ETL, QVT, Henshin, Viatra, …

Use case: A platform for analysing a social network

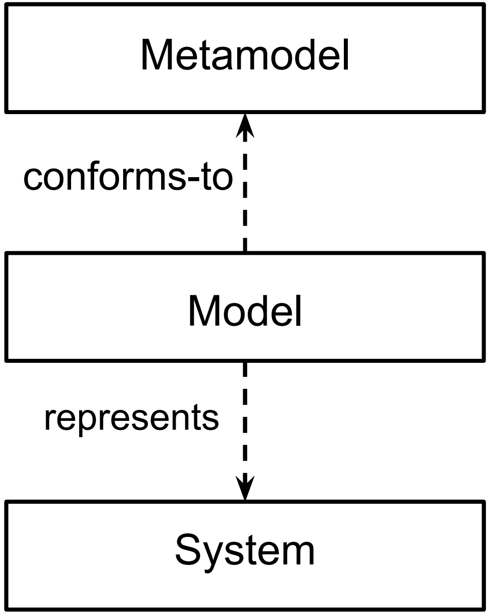


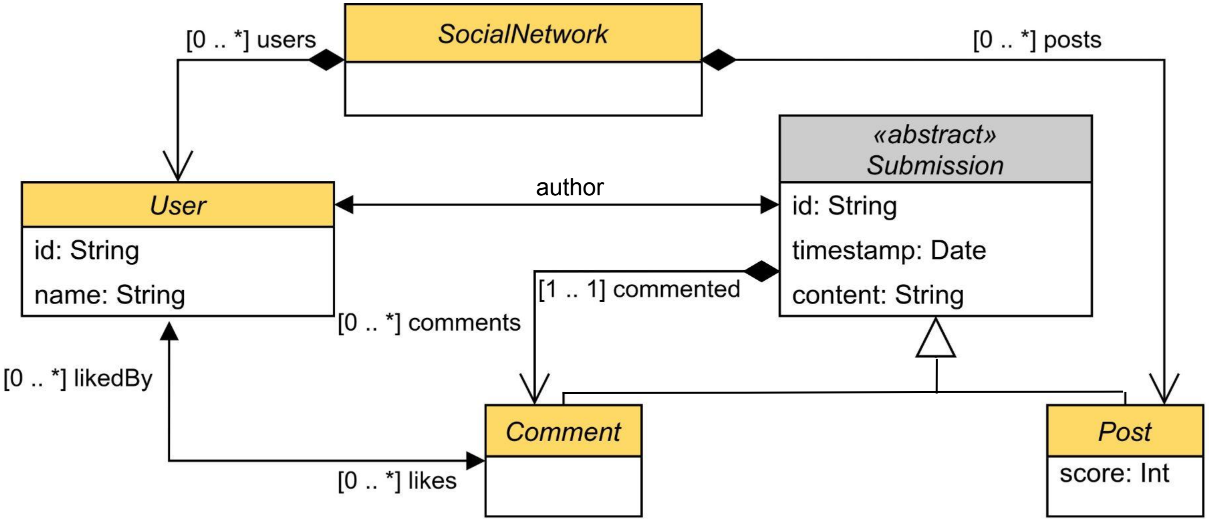
Use case: A platform for analysing a social network



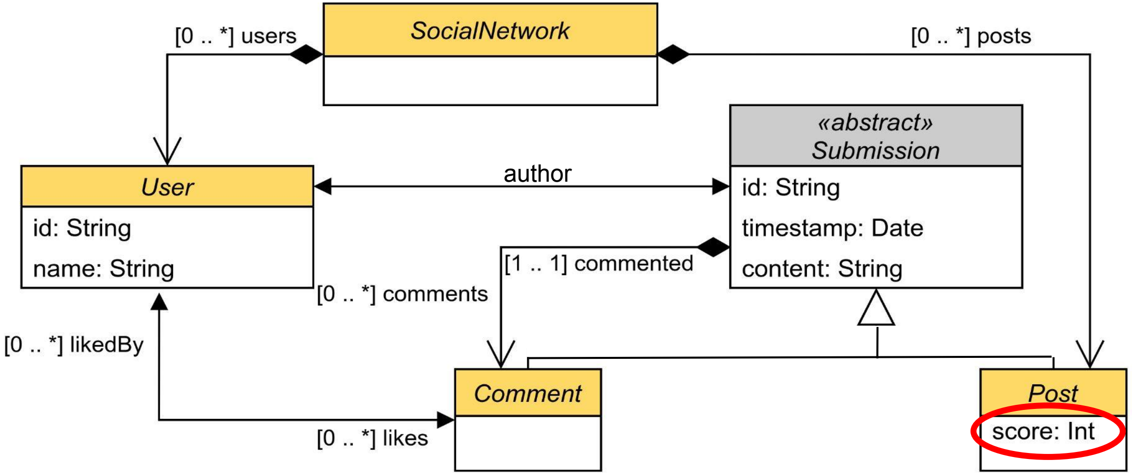


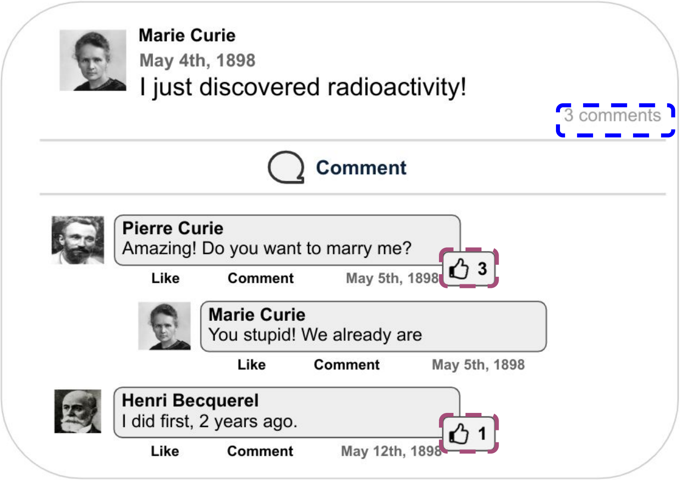
Use case: A platform for analysing a social network





Example 1: Give an activity score for posts in a social network

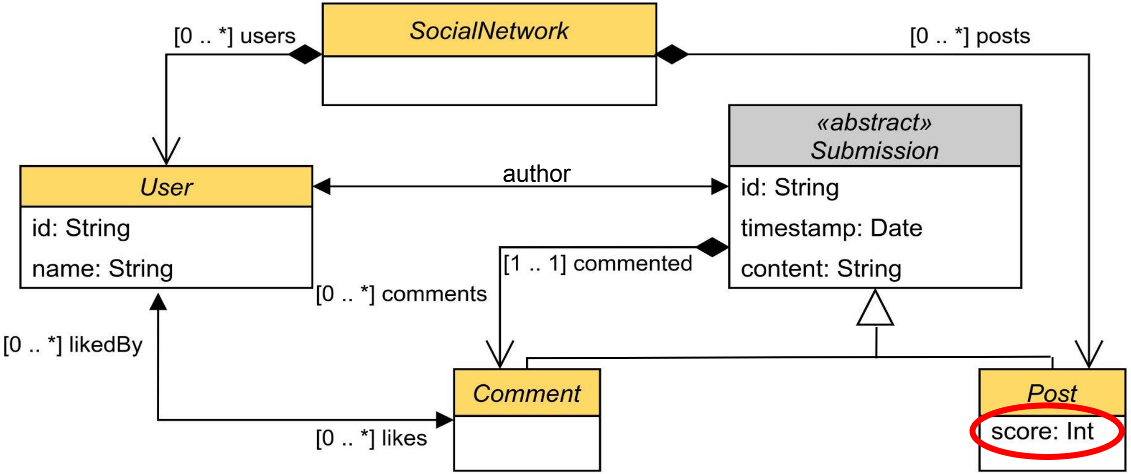


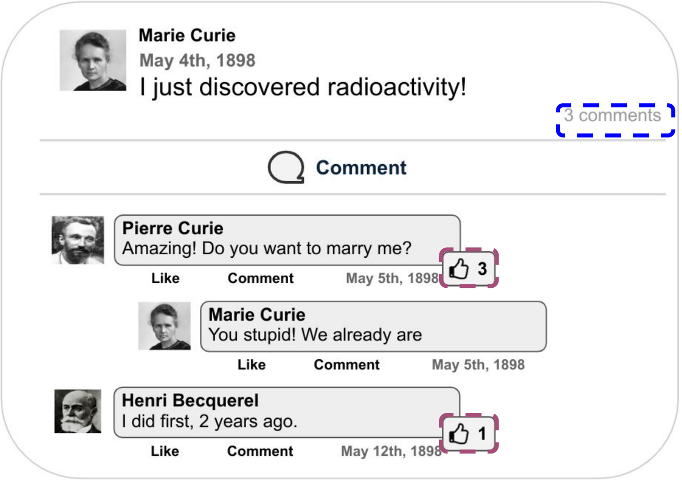


Example: score(p: *Post*) ≔ # comments × 10 + # likes

score(*Radioactivity*) = 3 × 10 + 4 = 34

Example 1: Give an activity score for posts in a social network



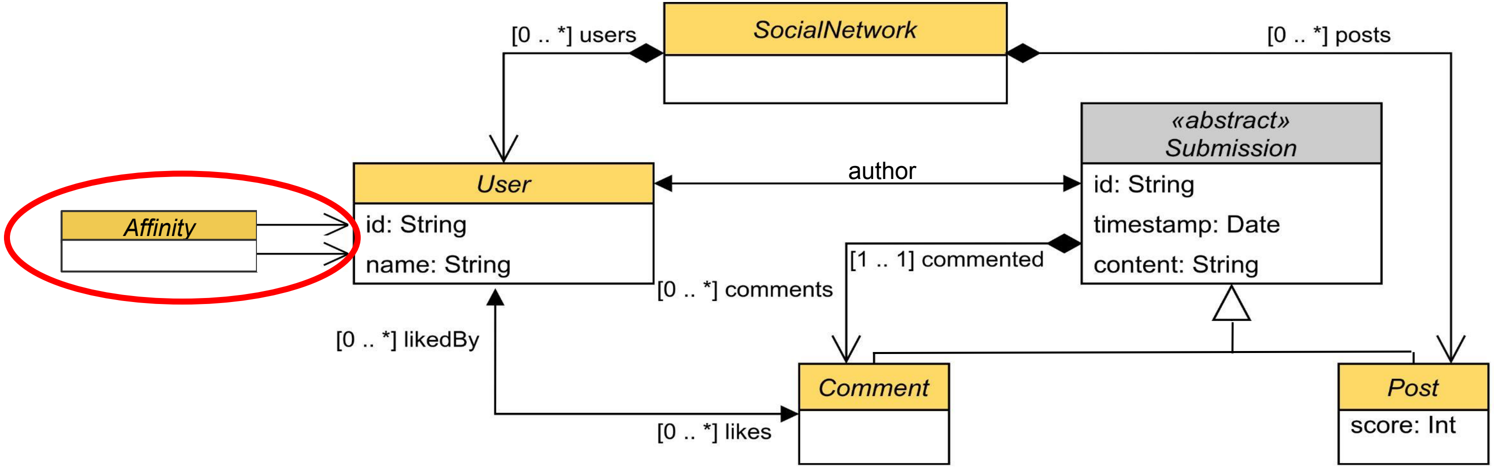


Example: score(p: *Post*) ≔ **# comments** × 10 + **# likes**

```
rule Post2ScoredPost (p:Post)
output:
  new Post (id ← p.id,
            timestamp ← p.timestamp
            content ← p.content,
            score ← score(p))
```

score as a **query**

Example 2: Look for user affinities in a social network

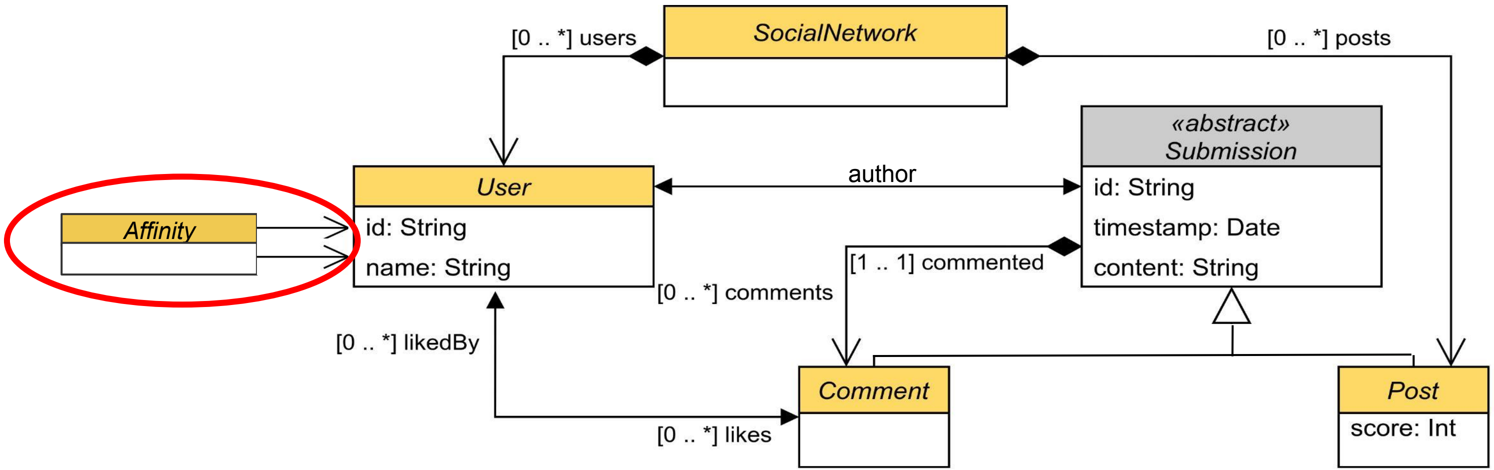


Example: Comment at least 3 same posts

Pierre Curie

Marie Curie

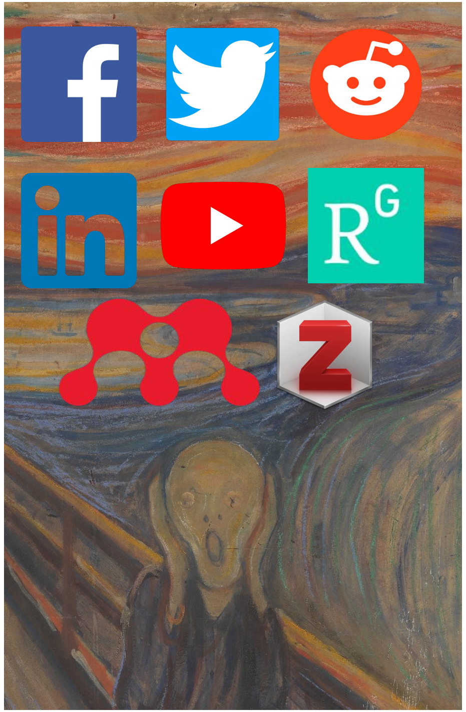
Example 2: Look for user affinities in a social network



Example: Comment at least 3 same posts

rule FindAffinity (u_1:User, u_2:User)
matching:
$\text{commentedPosts}(u_1) \cap \text{commentedPosts}(u_2) \geqslant 3$
output:
new Affinity($user_1 \leftarrow u_1$, $user_2 \leftarrow u_2$)

Model management for Very Large Models (VMLs)[1]

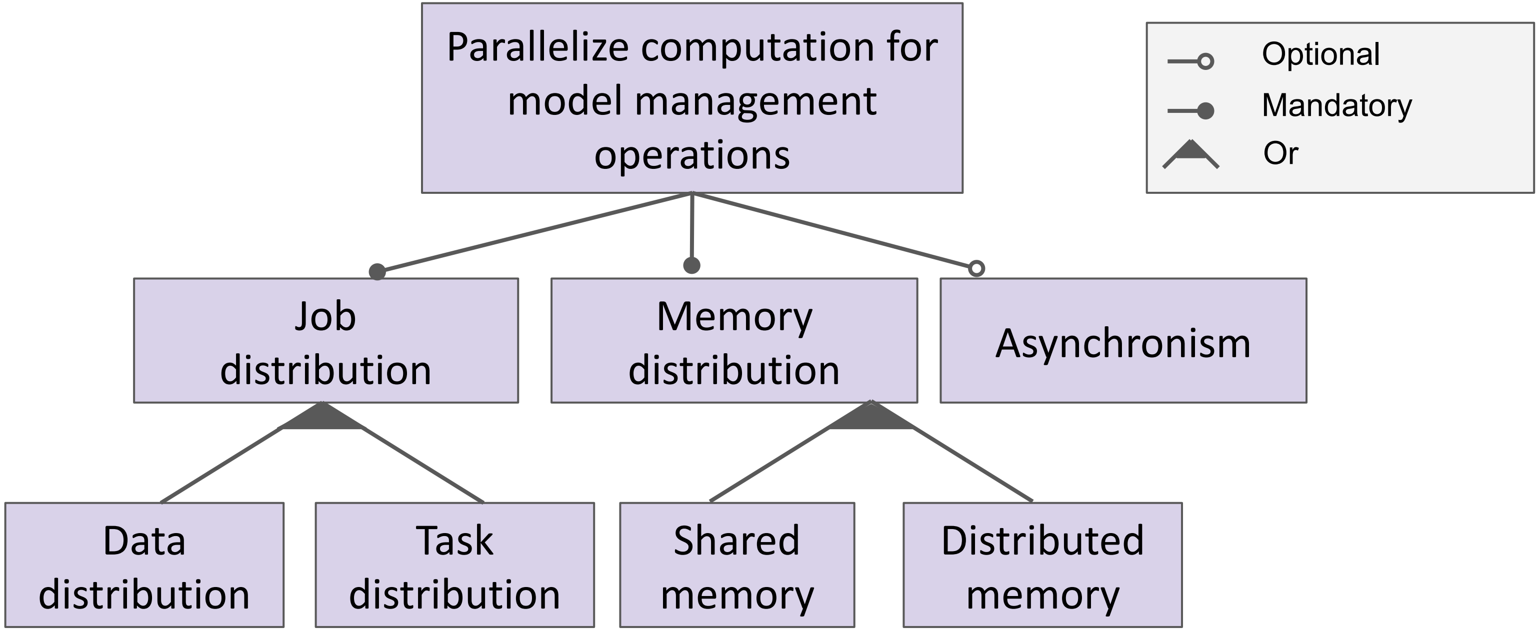


- Computational complexity
  - Size of the model
  - Storage and memory constraints
- Scalability with increasing resources
- Implicit optimization

➢ Two main approaches
  - Avoid computation
  - Parallelize computation

[1] Dimitrios S. Kolovos, Louis M. Rose, Nicholas Drivalos Matragkas, Richard F. Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan de Lara, István Ráth, Dániel Varró, Massimo Tisi, Jordi Cabot. **A research roadmap towards achieving scalability in model driven engineering.** *BigMDE@STAF 2013*

Scalability of model management for VLMs



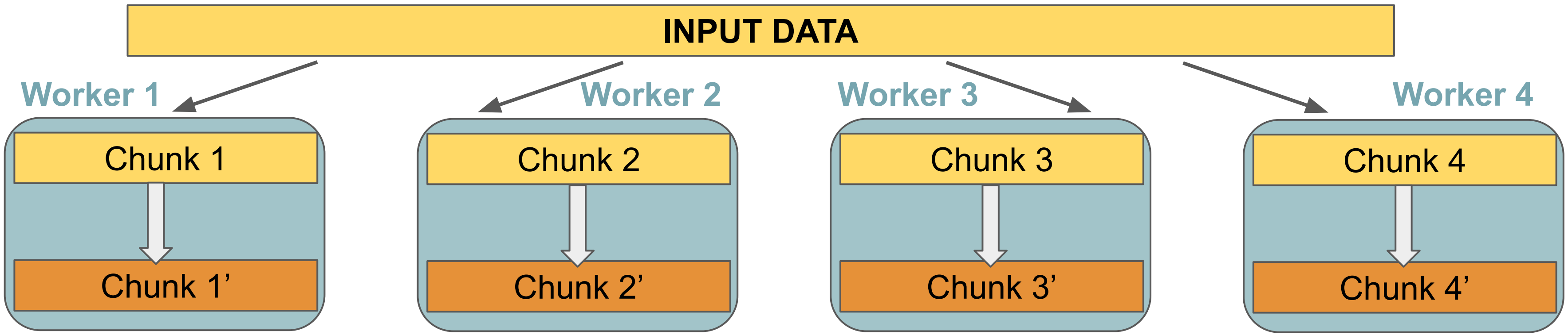
Parallelization in model transformation

	Model query	Model transfo.	Pattern match.	Optimization	Shared mem.	Distrib. mem.	Task-parallel	Data-parallel	Asynchronism
Amine Benelallam et al. «**Efficient model partitioning for distributed model** …» SLE 2016		X		X		X		X	
Amine Benelallam et al. «**ATL-MR: model transformation on MapReduce**» SPLASH 2015		X				X		X	
Loli Burgueño et al. «**A Linda-Based platform for the parallel execution** …» IST 2016		X			X			X	X
Loli Burgueño et al. «**Towards distributed model transformations with LinTra**» JISBD 2016		X		X		X		X	X
Loli Burgueño et al. «**Parallel in-place model transformations with LinTra**» CEUR-WS 2015		X			X		X		X
Jesús S. Cuadrado et al. «**Efficient execution of ATL model transformations** …» TSE 2020		X			X			X	
Gábor Imre et al. «**Parallel graph transformations on multicore systems**» MSEPT 2012		X			X		X		
Christian Krause et al. «**Implementing graph transformations in the BSP model**» FASE 2014			X			X		X	
Sina Madani et al. «**Distributed model validation with Epsilon**» SSM 2021	X				X	X		X	
Sina Madani et al. «**Towards optimisation of model queries: a parallel** …» ECMFA 2019	X			X	X		X		
Gergely Mezei et al. «**Towards truly parallel model transformations: a** …» EURCON 2019			X			X	X		
Massimo Tisi et al. «**Parallel execution of ATL transformation rules**» MODELS 2013		X			X		X		
Le-Duc Tung et al. «**Towards systematic parallelization of graph transfo.** …» IJPP 2017		X				X		X	
Tamás Vajk et al. «**Runtime model validation with parallel object** …» MoDeVVa 2011	X				X		X		

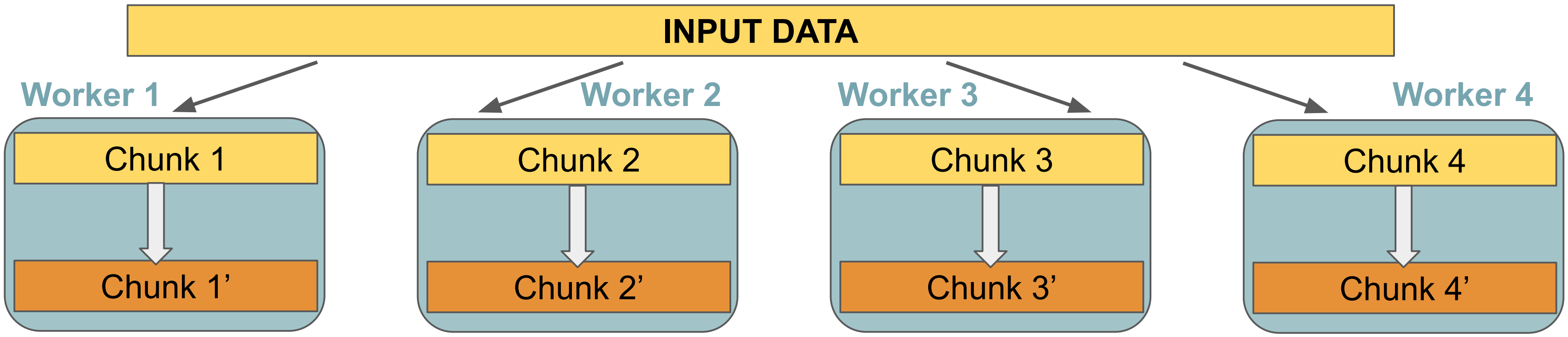
Parallelization in model transformation

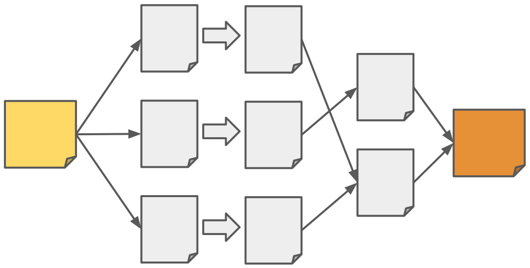
	Model query	Model transfo.	Pattern match.	Optimization	Shared mem.	Distrib. mem.	Task-parallel	Data-parallel	Asynchronism
Amine Benelallam et al. «Efficient model partitioning for distributed model …» SLE 2016		X		X		X		X	
Amine Benelallam et al. «ATL-MR: model transformation on MapReduce» SPLASH 2015		X				X		X	
Loli Burgueño et al. «A Linda-Based platform for the parallel execution …» IST 2016		X			X			X	X
Loli Burgueño et al. «Towards distributed model transformations with LinTra» JISBD 2016		X		X		X		X	X
Loli Burgueño et al. «Parallel in-place model transformations with LinTra» CEUR-WS 2015		X			X		X		X
Jesús S. Cuadrado et al. «Efficient execution of ATL model transformations …» TSE 2020		X			X			X	
Gábor Imre et al. «Parallel graph transformations on multicore systems» MSEPT 2012		X			X		X		
Christian Krause et al. «Implementing graph transformations in the BSP model» FASE 2014			X			X		X	
Sina Madani et al. «Distributed model validation with Epsilon» SSM 2021	X				X	X		X	
Sina Madani et al. «Towards optimisation of model queries: a parallel …» ECMFA 2019	X			X	X		X		
Gergely Mezei et al. «Towards truly parallel model transformations: a …» EURCON 2019			X			X	X		
Massimo Tisi et al. «Parallel execution of ATL transformation rules» MODELS 2013		X			X		X		
Le-Duc Tung et al. «Towards systematic parallelization of graph transfo. …» IJPP 2017		X				X		X	
Tamás Vajk et al. «Runtime model validation with parallel object …» MoDeVVa 2011	X				X		X		

Data-parallelism as a strategy

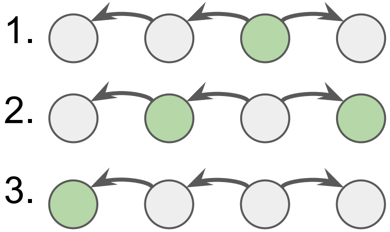


Data-parallelism as a strategy

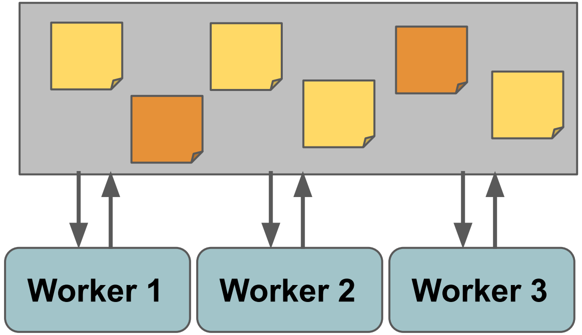




MapReduce



Pregel (vertex-centric)



Blackboard

Parallel / Distributed in model transformation

- Large number of distributed engines
  - Designed with ≠ **purposes**
  - Following ≠ **design choices**
  - Implemented on ≠ **languages** for ≠ **infrastructures**

⇒ What are the optimal design choices for a given case?

Related work

- Automatic adapted strategy
  - Pattern matching (**Bergman et al.**)
- Classification of features of MDE solution
  - For languages (**Tamura et al., M Rose et al.**)
  - Transformation approaches (**Czarnecki et al., Kahani et al.**)
  - Performance oriented (**Groner et al.**)
  - Specific topic: bi-directionality (**Hidaka et al.**)

Optimization in model transformation

	Model query	Model transfo.	Pattern match.	Optimization	Shared mem.	Distrib. mem.	Task-parallel	Data-parallel	Asynchronism
Amine Benelallam et al. «**Efficient model partitioning for distributed model** …» SLE 2016		X		X		X		X	
Amine Benelallam et al. «**ATL-MR: model transformation on MapReduce**» SPLASH 2015		X				X		X	
Loli Burgueño et al. «**A Linda-Based platform for the parallel execution** …» IST 2016		X			X			X	X
Loli Burgueño et al. «**Towards distributed model transformations with LinTra**» JISBD 2016		X		X		X		X	X
Loli Burgueño et al. «**Parallel in-place model transformations with LinTra**» CEUR-WS 2015		X			X		X		X
Jesús S. Cuadrado et al. «**Efficient execution of ATL model transformations** …» TSE 2020		X			X			X	
Gábor Imre et al. «**Parallel graph transformations on multicore systems**» MSEPT 2012		X			X		X		
Christian Krause et al. «**Implementing graph transformations in the BSP model**» FASE 2014			X			X		X	
Sina Madani et al. «**Distributed model validation with Epsilon**» SSM 2021	X				X	X		X	
Sina Madani et al. «**Towards optimisation of model queries: a parallel** …» ECMFA 2019	X			X	X		X		
Gergely Mezei et al. «**Towards truly parallel model transformations: a** …» EURCON 2019			X			X	X		
Massimo Tisi et al. «**Parallel execution of ATL transformation rules**» MODELS 2013		X			X		X		
Le-Duc Tung et al. «**Towards systematic parallelization of graph transfo.** …» IJPP 2017		X				X		X	
Tamás Vajk et al. «**Runtime model validation with parallel object** …» MoDeVVa 2011	X				X		X		

Problem: A configuration issue

- What solution to use?
- How to optimally configure a solution?

Problem 1:
Many solutions for executing rules distributively

Problem 2:
Many solutions for executing queries distributively

Problem 3:
Lack of unified proposition for comparing design choices

➢ **Goal:** Getting an insight of how design choices impact scalability of a distributed transformation

Contribution of the thesis

Problem 1:
Many solutions for executing rules distributively

Evaluation of distributed design choices for **rule execution**

- Building a new distributed transformation engine: SparkTE

Contribution of the thesis

Problem 1:
Many solutions for executing rules distributively

Evaluation of distributed design choices for **rule execution**

- Building a new distributed transformation engine: SparkTE

Problem 2:
Many solutions for executing queries distributively

Evaluation of distributed design choices for **query execution**

- Analysing different distributed execution strategies for a query

Contribution of the thesis

Problem 1:
Many solutions for executing rules distributively

Evaluation of distributed design choices for **rule execution**

- Building a new distributed transformation engine: SparkTE

Problem 2:
Many solutions for executing queries distributively

Evaluation of distributed design choices for **query execution**

- Analysing different distributed execution strategies for a query

Problem 3:
Lack of unified proposition for comparing design choices

Make possible configurable distributed transformation

- Modeling the design space
- Making the configurable engine: Configurable SparkTE

1st Contribution

2.1 SPARKTE, A DISTRIBUTED MODEL TRANSFORMATION ENGINE

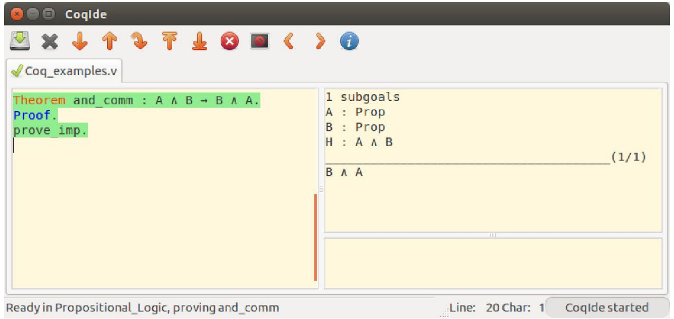
Problematic and proposal

Many solutions for executing rules distributively

- ➢ Evaluation of distributed design choices for **rule execution**
  - An engine with design choices for rule execution: SparkTE
  - Prove design choices have no impact on the result
  - Evaluate the scalability of a such engine

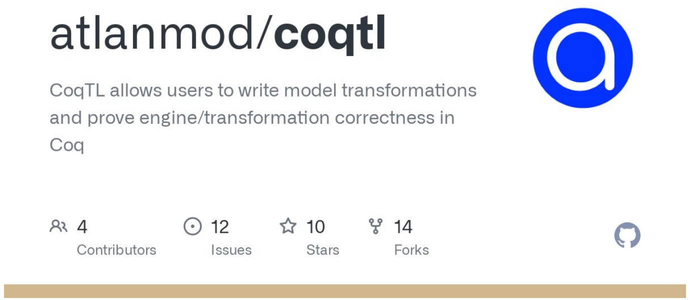
CoqTL for reasoning

Coq

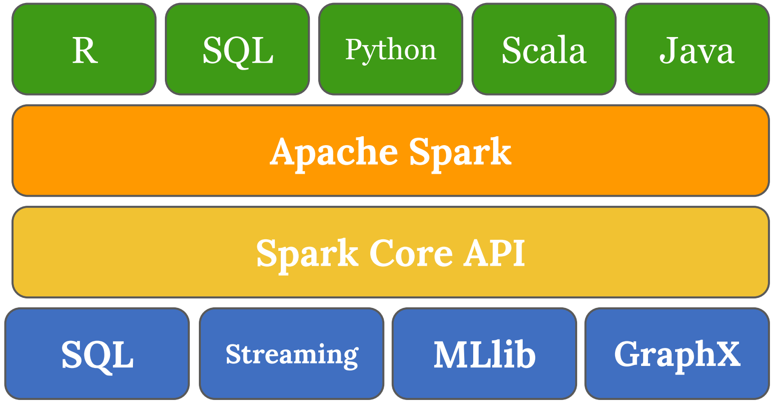


- Designed for specifying semantics
- A proof assistant based on **Hoare logic**
- Extraction mechanism (to ML lang)

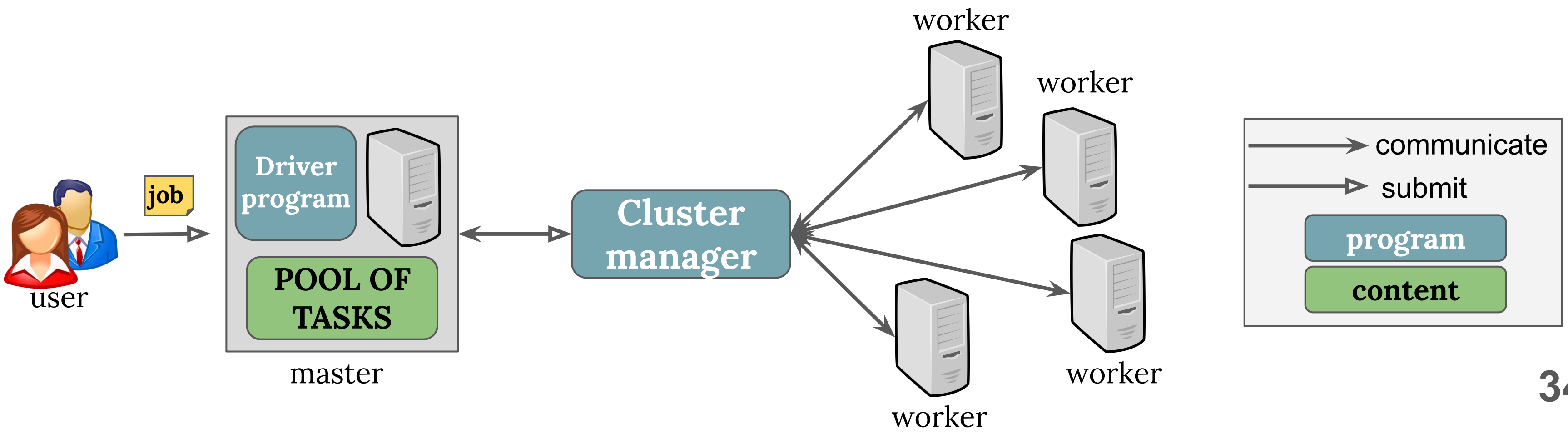
- **DSL** for rule-based model transformation
- Made for **reasoning on transformations**
- Can **reason on the semantic** of the transformation



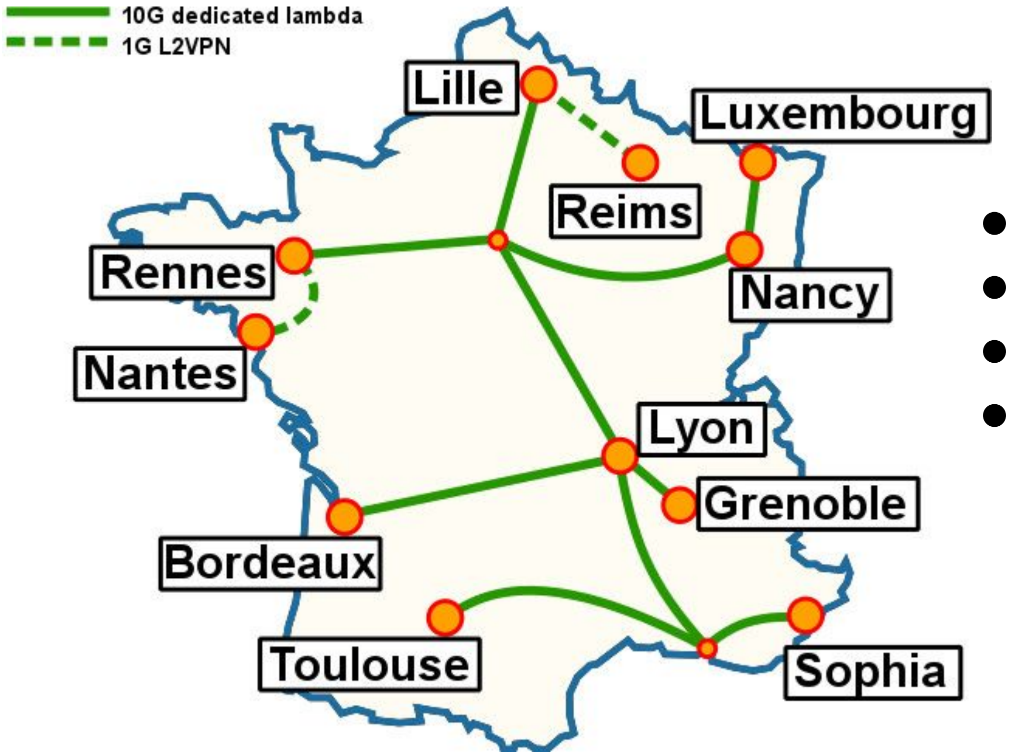
Spark as a target



- Popular distributed computing for **large-scale data** processing
- Support for **many paradigms**
  - MapReduce, vertex-based (Pregel), …
- **Open-source**

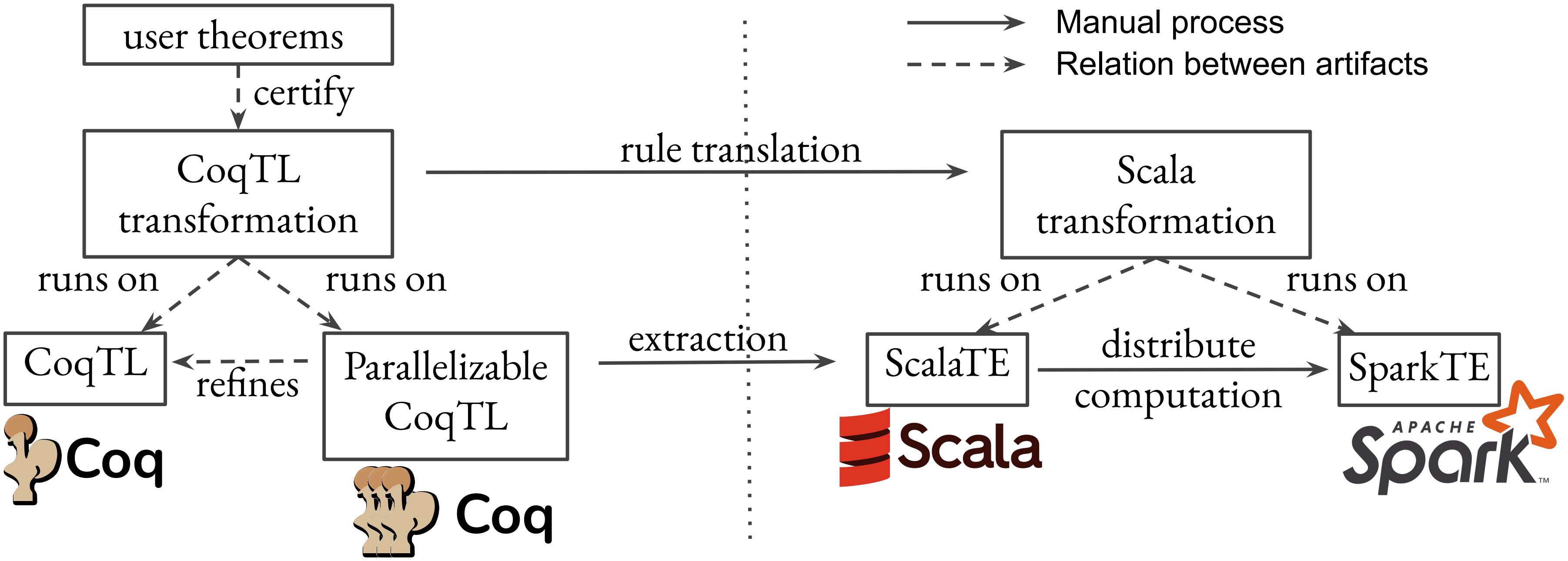


Spark on Grid'5000

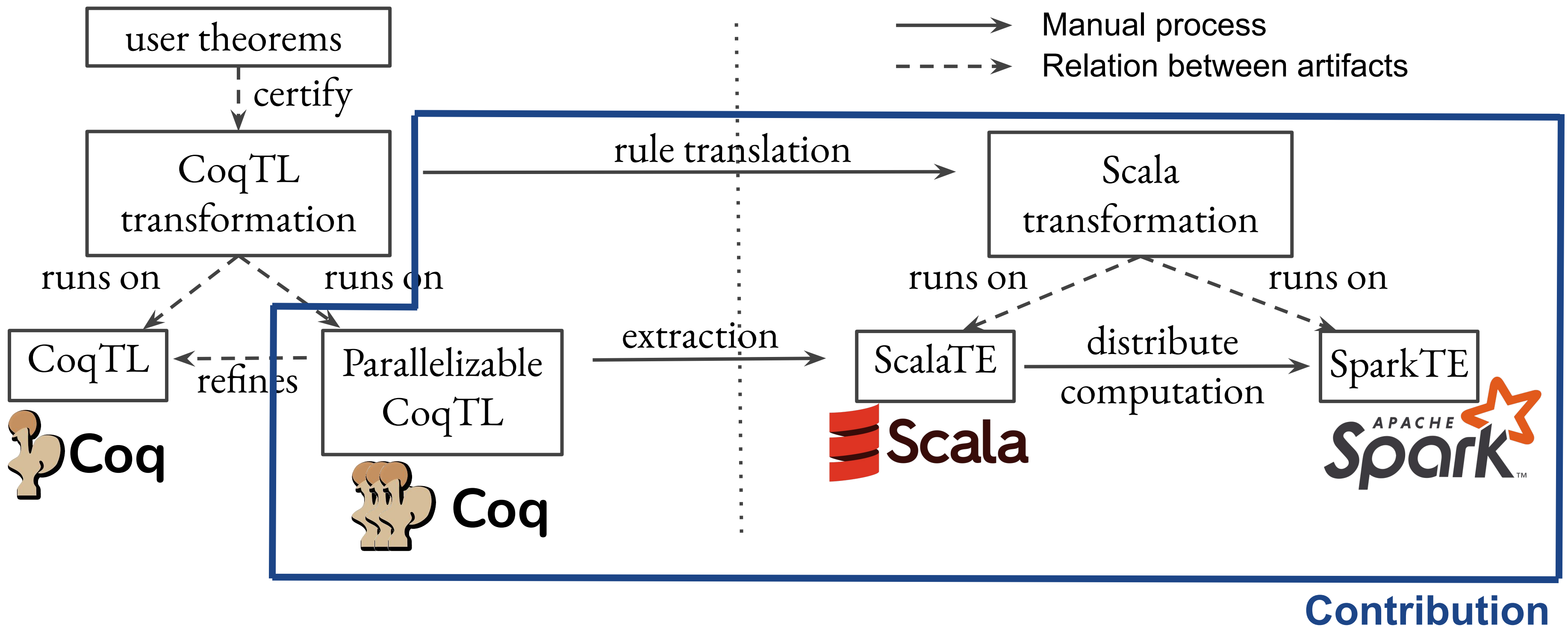


- French cluster for experimentation
- Library for benchmarking
- Support for distributed computing
- More than 15,000 cores; 800 nodes

Engine based on a formal semantic: from CoqTL to SparkTE



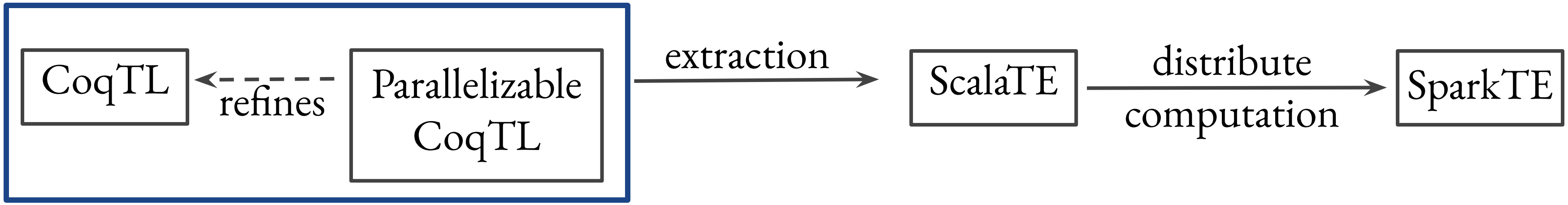
Engine based on a formal semantic: from CoqTL to SparkTE



Contribution: Parallelizable CoqTL as a CoqTL refinement

- **Increase parallelization**
  1. Two distinct phases: **instantiate** & **apply**
     - Define map-reduce phases
  2. Iterate on rules instead of src patterns
     - Avoid unnecessary computations
  3. Iterate on trace links instead of src patterns
     - Reuse of intermediate results

- **Formal proof** of equivalence with CoqTL

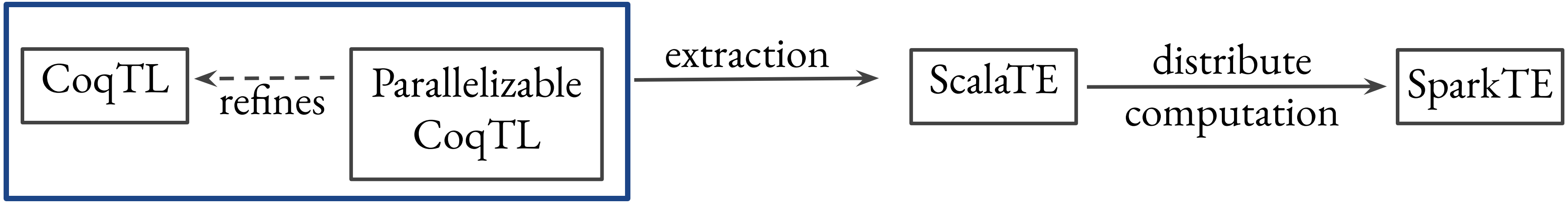


Contribution: Parallelizable CoqTL as a CoqTL refinement

- **Increase parallelization**
  1. Two distinct phases: **instantiate** & **apply**
     - Define map-reduce phases
  2. Iterate on rules instead of src patterns
     - Avoid unnecessary computations
  3. Iterate on trace links instead of src patterns
     - Reuse of intermediate results

	Spec. size (LoC)	Cert. size (LoC)	Proof effort (man-days)
1.	69	484	10
2.	42	487	7
3.	69	520	4

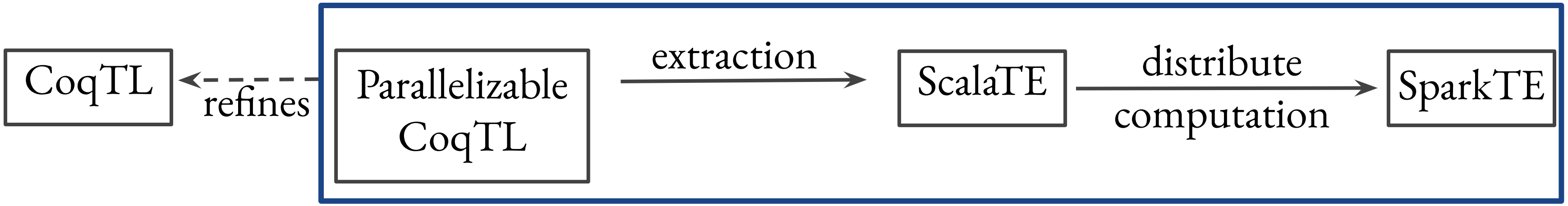
- **Formal proof** of equivalence with CoqTL



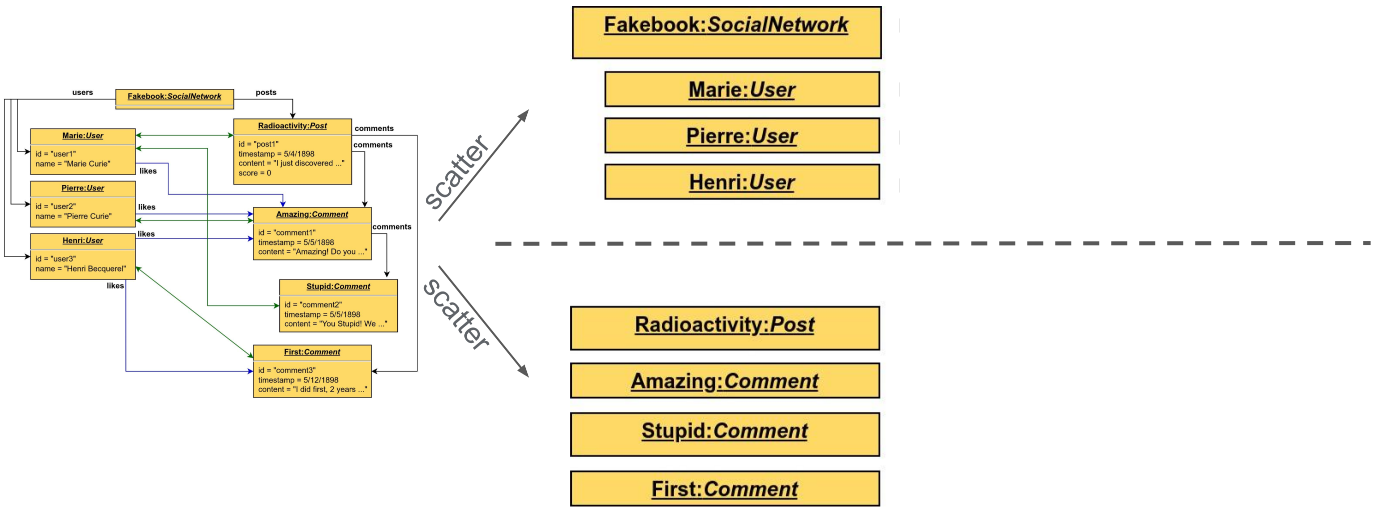
Contribution: Build executable and distributed transformation engine

1. **Produce** executable and maintainable **code**
   - Object-oriented approach
   - Pure Scala functions (correctness)

2. **Distribute the computation**
   - Distribute data-structures
   - Explicit communication operations
     - Take advantage of scatter/gather operations
     - Broadcast global knowledge



Instantiate phase: Create output elements



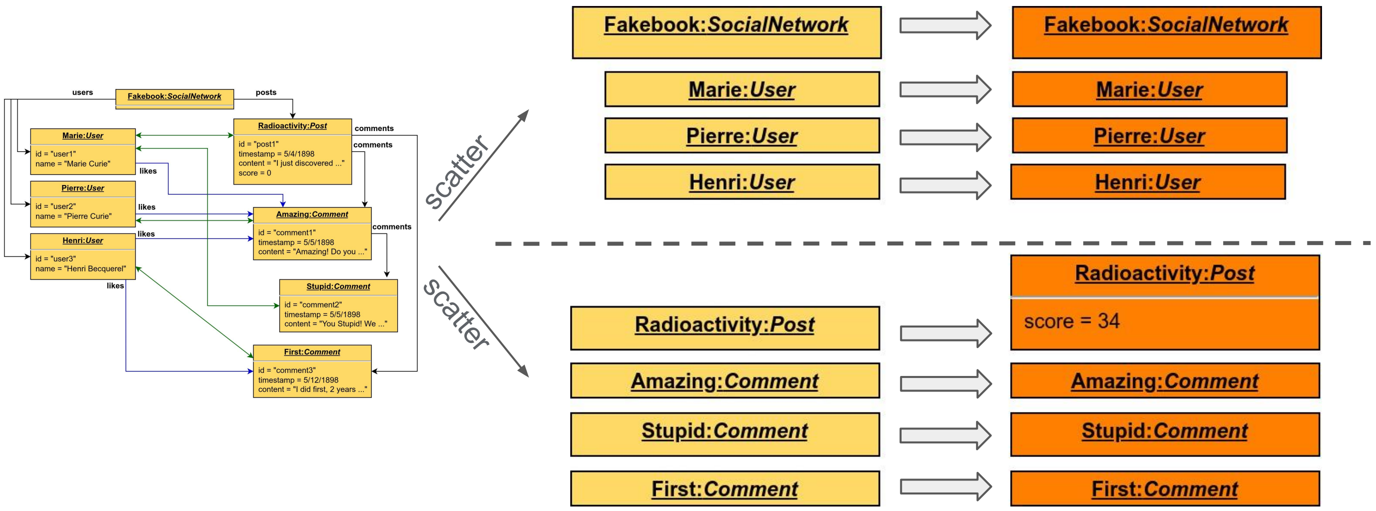
Data-distributed strategy: (*Map-Reduce* phase)

- Input **elements** are **distributed**
- Input **model** is **broadcasted**

As output:

- Instantiated **output model elements**
- **Trace-links** (mapping input-output)

Instantiate phase: Create output elements



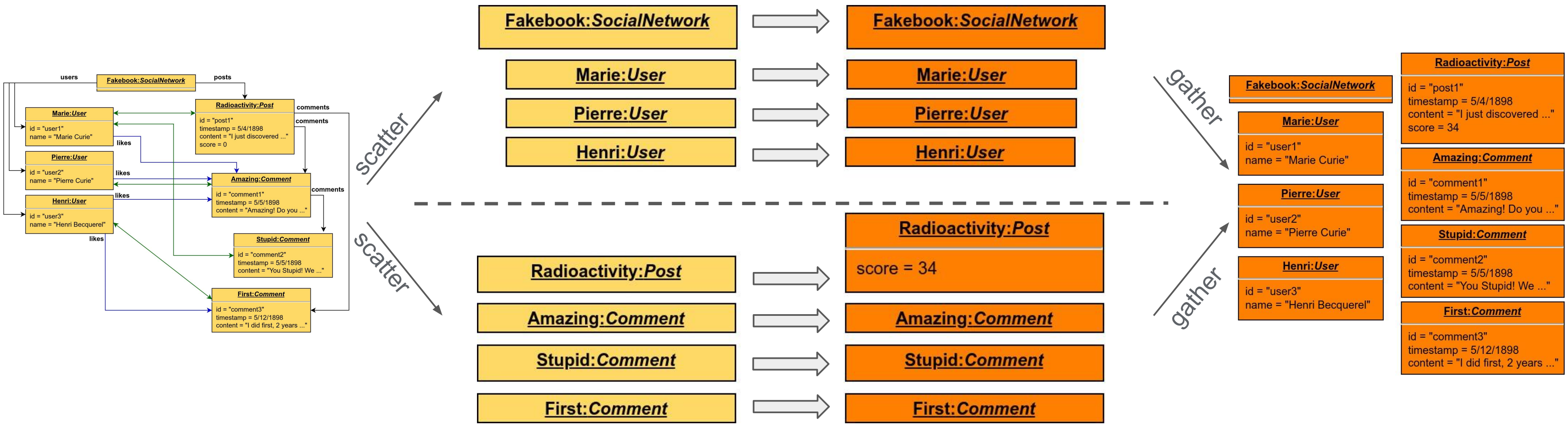
Data-distributed strategy: (*Map-Reduce* phase)

- Input **elements** are **distributed**
- Input **model** is **broadcasted**

As output:

- Instantiated **output model elements**
- **Trace-links** (mapping input-output)

Instantiate phase: Create output elements



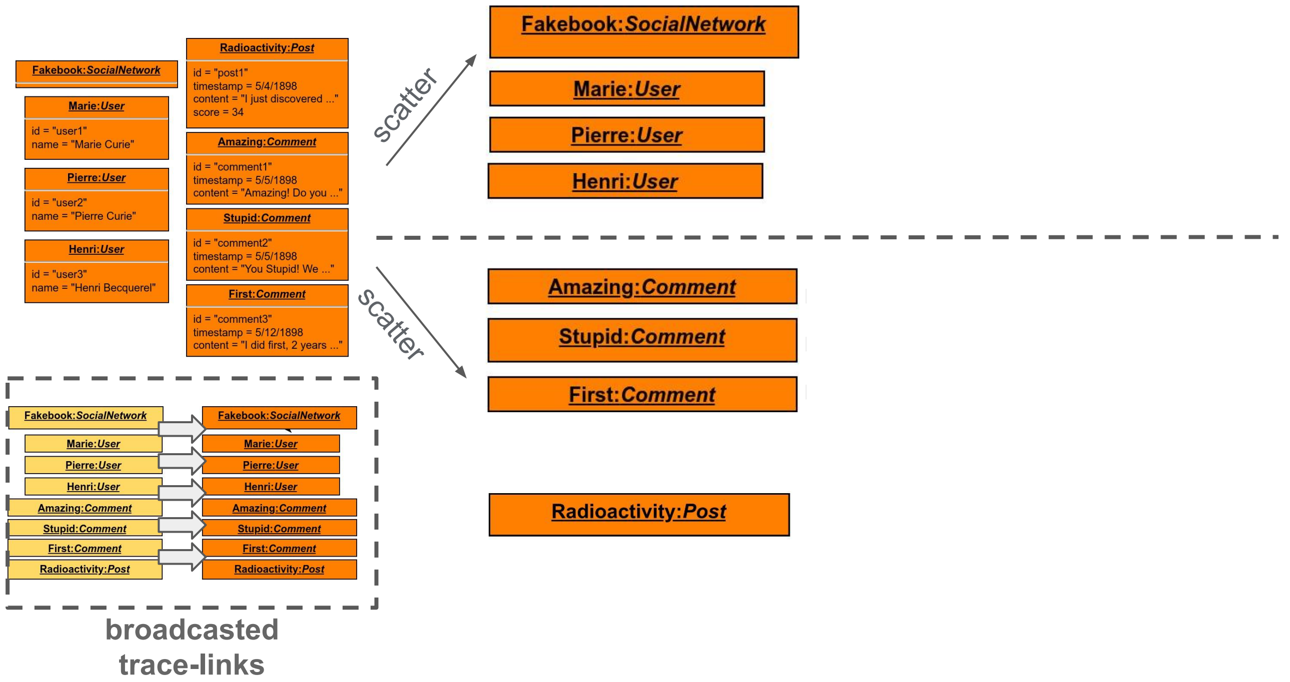
Data-distributed strategy: (*Map-Reduce* phase)

- Input **elements** are **distributed**
- Input **model** is **broadcasted**

As output:

- Instantiated **output model elements**
- **Trace-links** (mapping input-output)

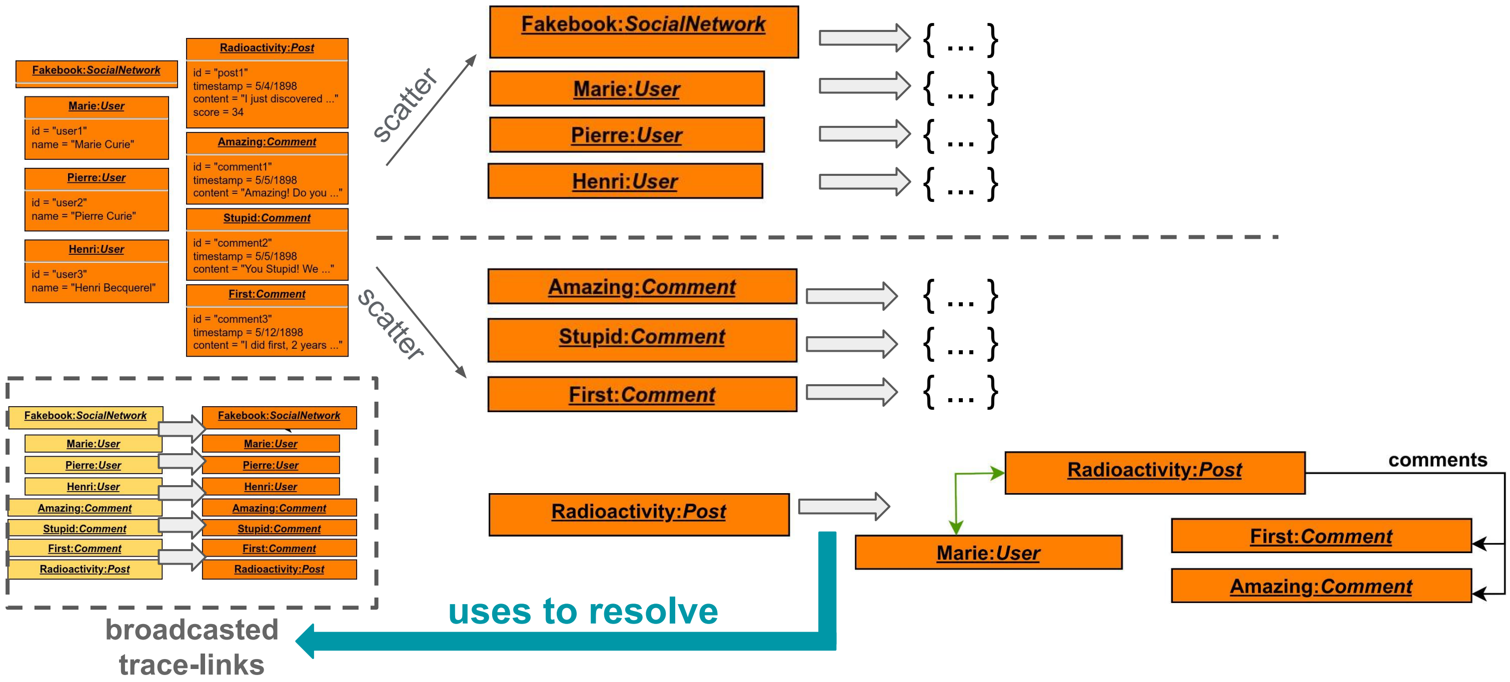
Apply phase: Create output links



Data-distributed strategy: (*Map-Reduce* phase)

- Output **elements** are **distributed**
- Trace-links are **broadcasted**

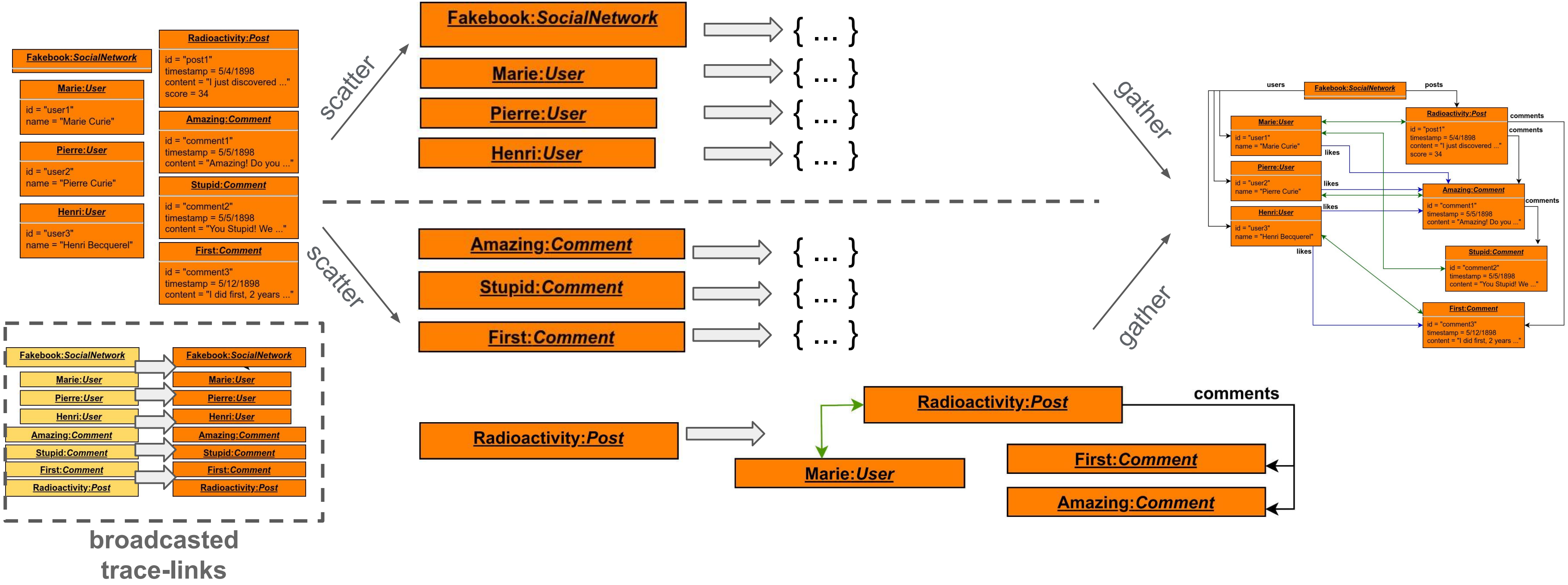
Apply phase: Create output links



Data-distributed strategy: (*Map-Reduce* phase)

- Output **elements** are **distributed**
- Trace-links are **broadcasted**

Apply phase: Create output links



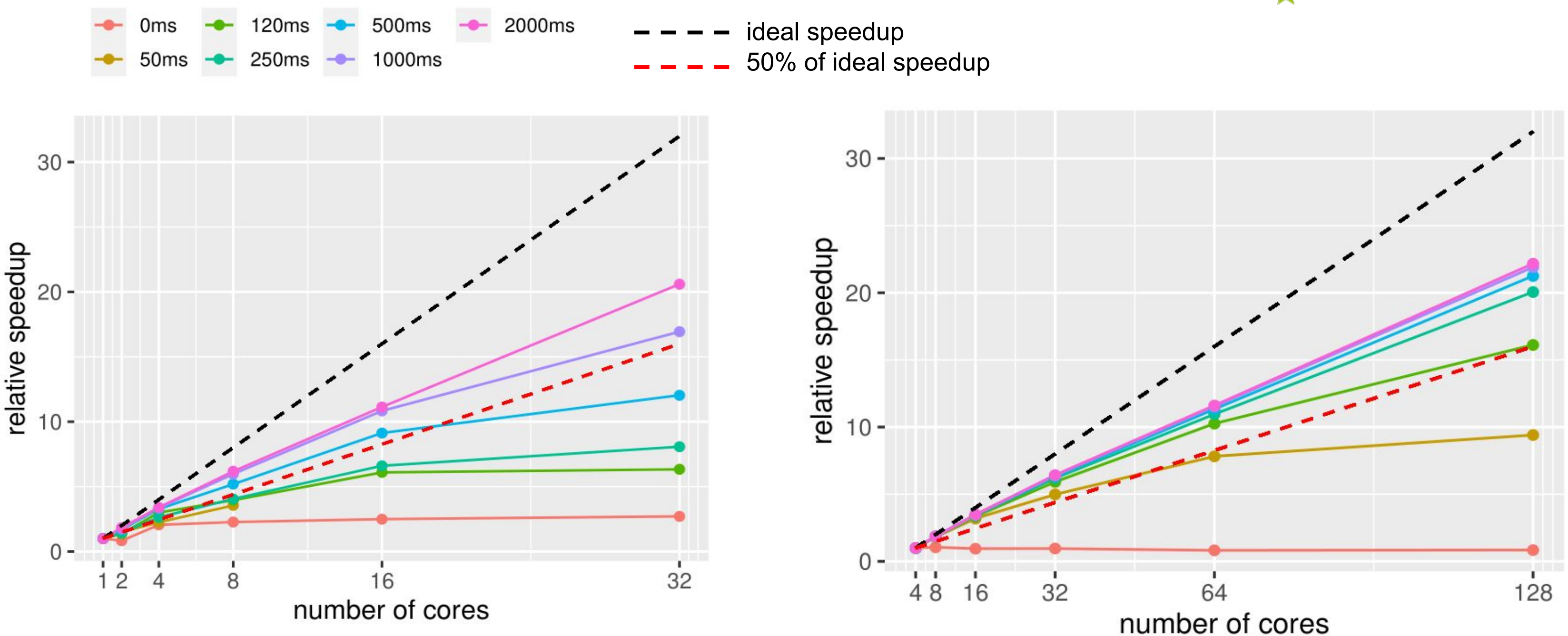
Data-distributed strategy: (*Map-Reduce* phase)

- Output **elements** are **distributed**
- Trace-links are **broadcasted**

Vertical scalability of model transformation on Spark

Grid'5000

- Simulate a uniform amount of computation on nodes
  - fixed time for each task



Model of 150 elements and 290 links, on 4 machines

Model of 600 elements and 1060 links, 8 machines

2nd Contribution

2.2 DISTRIBUTED QUERY EVALUATION STRATEGIES

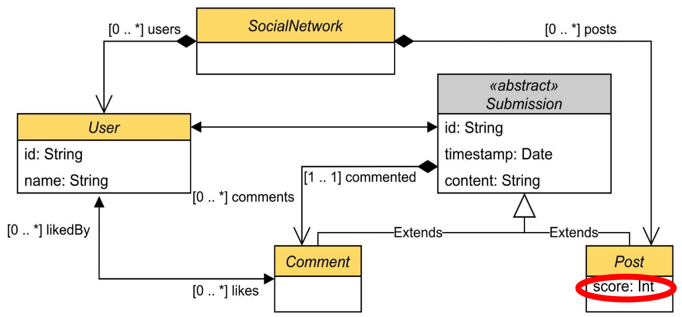
Problematic and proposal

Many solutions for executing queries distributively

- Evaluation of distributed design choices for **query execution**
  - Take a query whose evaluation is dependant from input model
  - Implement with several design choices
  - Evaluate them and try to correlate with input

Several design choices for running a query

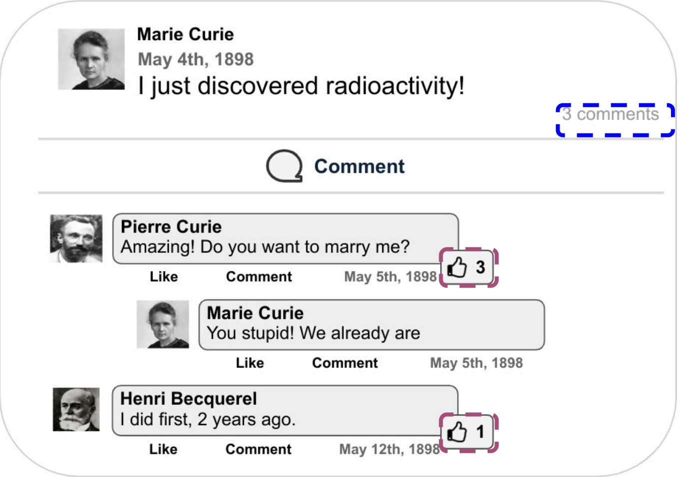
- Query:
  What is the score for a post in a social network?
- A score function
  score(p: *Post*) ≔ **# comments** × 10 + **# likes**




```
score(p: Post) :=   comments(p).size() * 10
                        + likes(p).size()

comments(s: Submission) :=  [s.comments].
        union(c: s.comments.flatMap(
                    λc.comments(c))

likes(p: Post) := comments(p).map(λc.likes)
```



Several design choices (implementation)

```
score(p: Post) :=
   comments(p).size() * 10
        + likes(p).size()

comments(s: Submission) :=
  [s.comments].union(
       c: s.comments.flatMap(
       λc.comments(c))

likes(p: Post) :=
  comments(p).map(λc.likes)
```

- Design-choices for running the query:
  1. **Scala-OCL**
     - No distribution (sequential)
  2. **Spark-OCL** (Spark core API)
     - Delegate distribution to Spark
  3. **MapReduce** (Spark core API)
     - More control of parallelism
  4. **Pregel** from (GraphX)
     - Iterative process
  5. Hybrid approaches
     - **Spark-OCL + Pregel**
     - **MapReduce + Pregel**

Experiments

- Proposed models from TTC
- Calculate score value
- Cannot really extract relevant metrics about topology

#	Dataset				Speed-up (compared to Sequential Scala-OCL)					
#	# users	# posts	# comments	# likes	Scala-OCL	Spark-OCL	Pregel	MapReduce	Spark-OCL + Pregel	MapReduce + Pregel
1	889	1064	118	24	**1x**	0.39x	0.36x	0.46x	0.44x	0.46x
2	1845	2315	190	66	**1x**	0.51x	0.68x	**0.85x**	0.66x	0.71x
3	2270	5056	204	129	**1x**	0.27x	0.35x	2.34x	0.15x	**2.96x**
4	5518	9220	394	572	**1x**	4.25x	**5.21x**	4.17x	4.68x	4.03x
5	10929	18872	595	1598	**1x**	**4.68x**	2.83x	2.39x	1.97x	3.91x
6	18083	39212	781	4770	**1x**	4.07x	4.12x	4.58x	**5.17x**	3.27x

Experiments: Correlation input vs. results

Correlation matrix: *input model* vs. *speed-ups*					
Size	**Spark- OCL**	**Pregel**	**MapReduce**	**Spark-OCL + Pregel**	**MapReduce + Pregel**
# users	0.78	0.67	0.74	0.76	0.39
# posts	0.71	0.62	0.75	0.75	0.32
# comments	**0.86**	**0.74**	**0.78**	**0.79**	**0.51**
# likes	0.62	0.57	0.7	0.73	0.19

Experiments: Correlation input vs. results

Correlation matrix: *input model* vs. *speed-ups*					
Size	**Spark- OCL**	**Pregel**	**MapReduce**	**Spark-OCL + Pregel**	**MapReduce + Pregel**
# users	0.78	0.67	0.74	0.76	0.39
# posts	0.71	0.62	0.75	0.75	0.32
# comments	**0.86**	**0.74**	**0.78**	**0.79**	**0.51**
# likes	0.62	0.57	0.7	0.73	0.19

Correlation matrix: *ratio in input model* vs *speed-ups*					
	Spark-OCL	**Pregel**	**MapReduce**	**Spark-OCL + Pregel**	**MapReduce + Pregel**
ratio: #users / #likes	-0.85	-0.79	**-0.89**	-0.75	-0.82
ratio: #posts / #likes	**-0.96**	**-0.88**	-0.82	**-0.85**	-0.66
ratio: #comments / #likes	-0.8	-0.74	-0.86	-0.69	**-0.83**

2.3

3rd Contribution

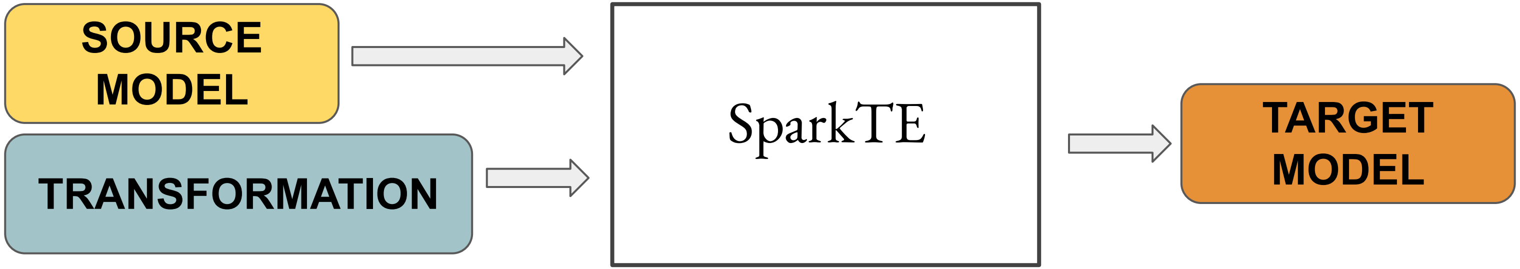
FEATURE ANALYSIS

Problematic and proposal

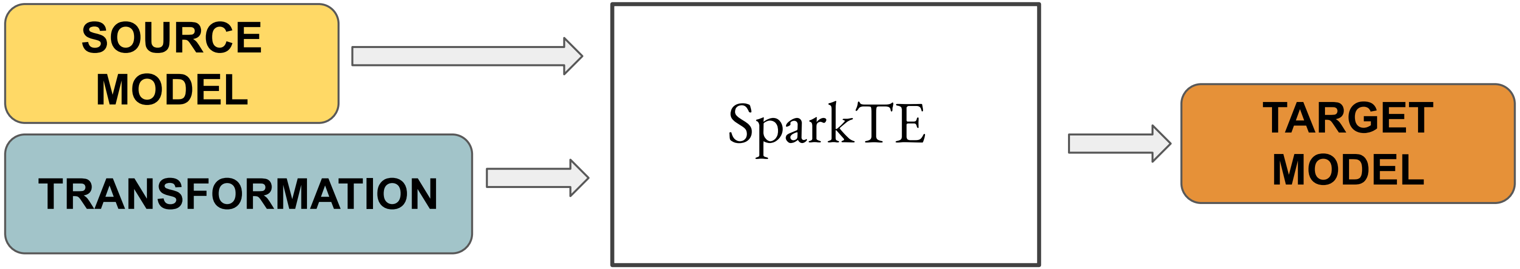
Lack of unified proposition for comparing design choices

- ➢ Make possible configurable distributed transformation
  - Formalized past contributions and additional design choices
  - Design a configurable engine
  - Evaluate them and analyse impact

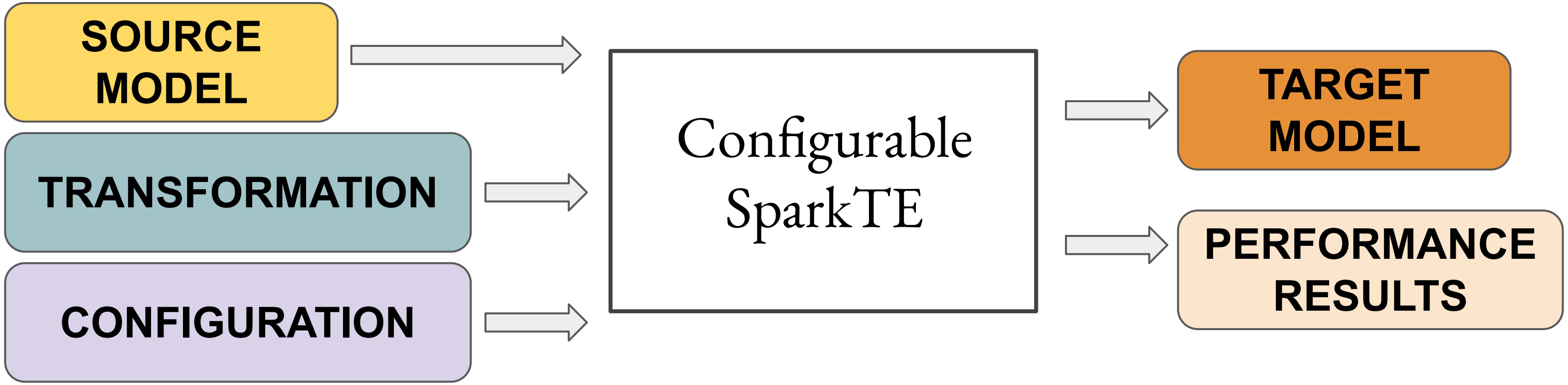
Configurable engine



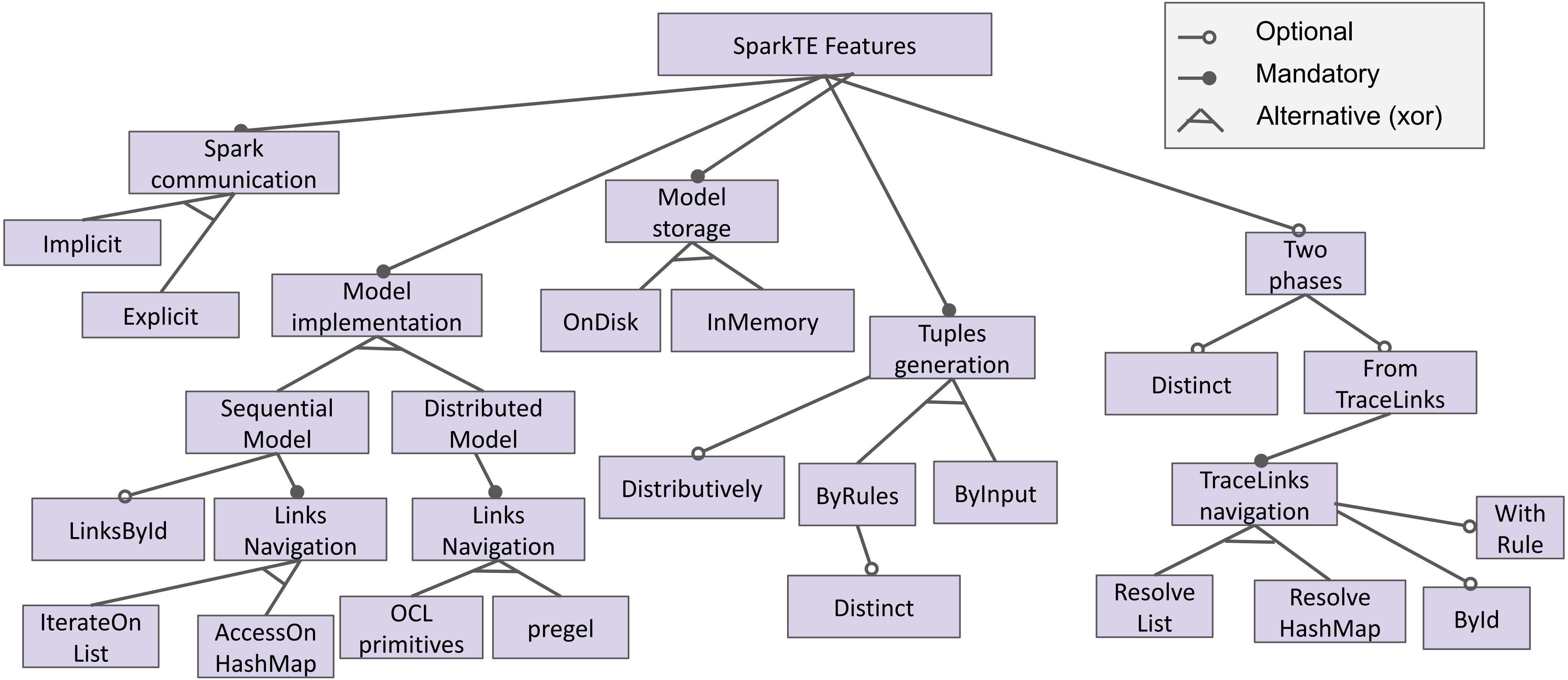
Configurable engine



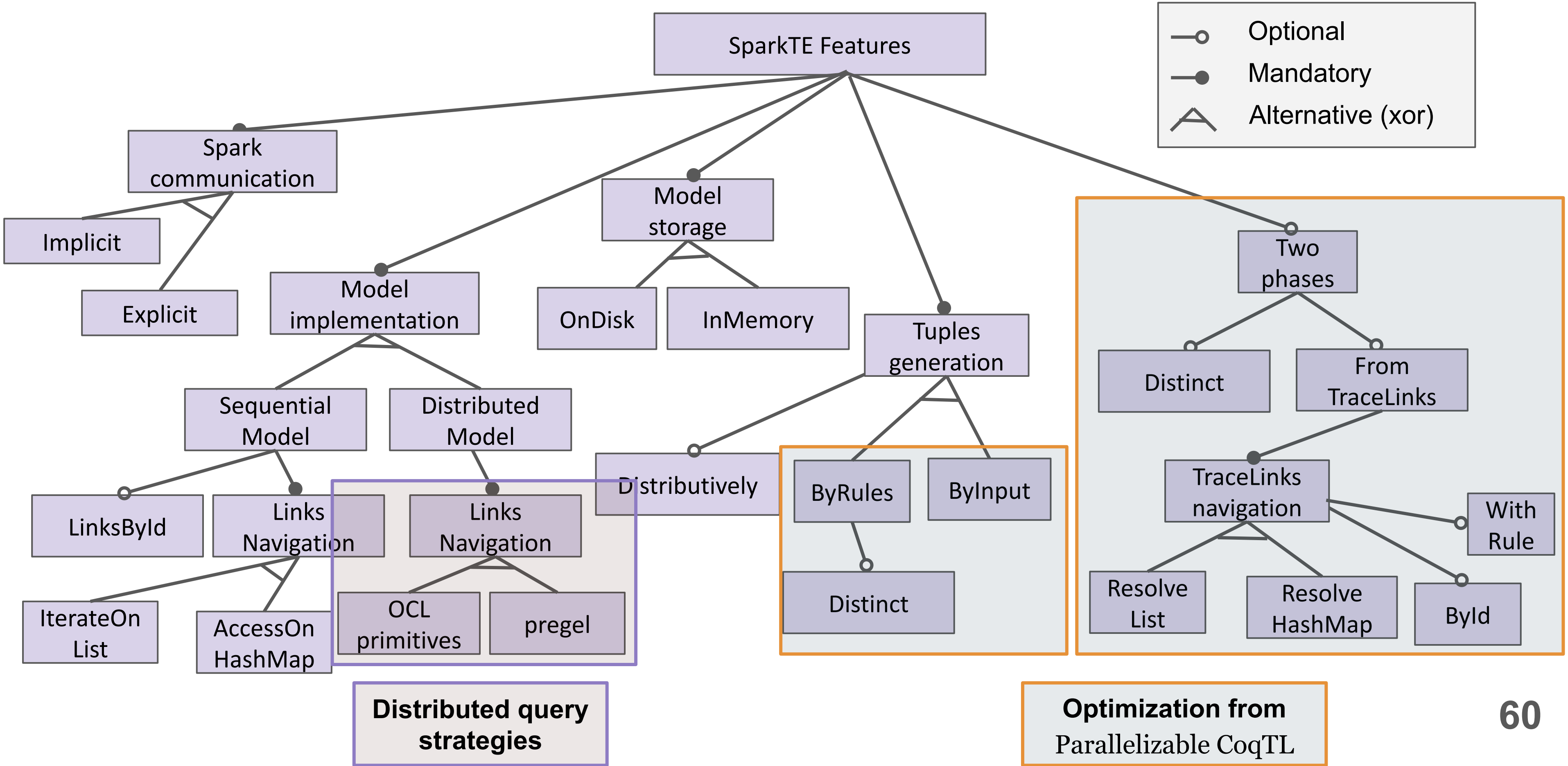
- Take **as input** a configuration conforms to the feature model
- Produce **as output** performance results (computation time)



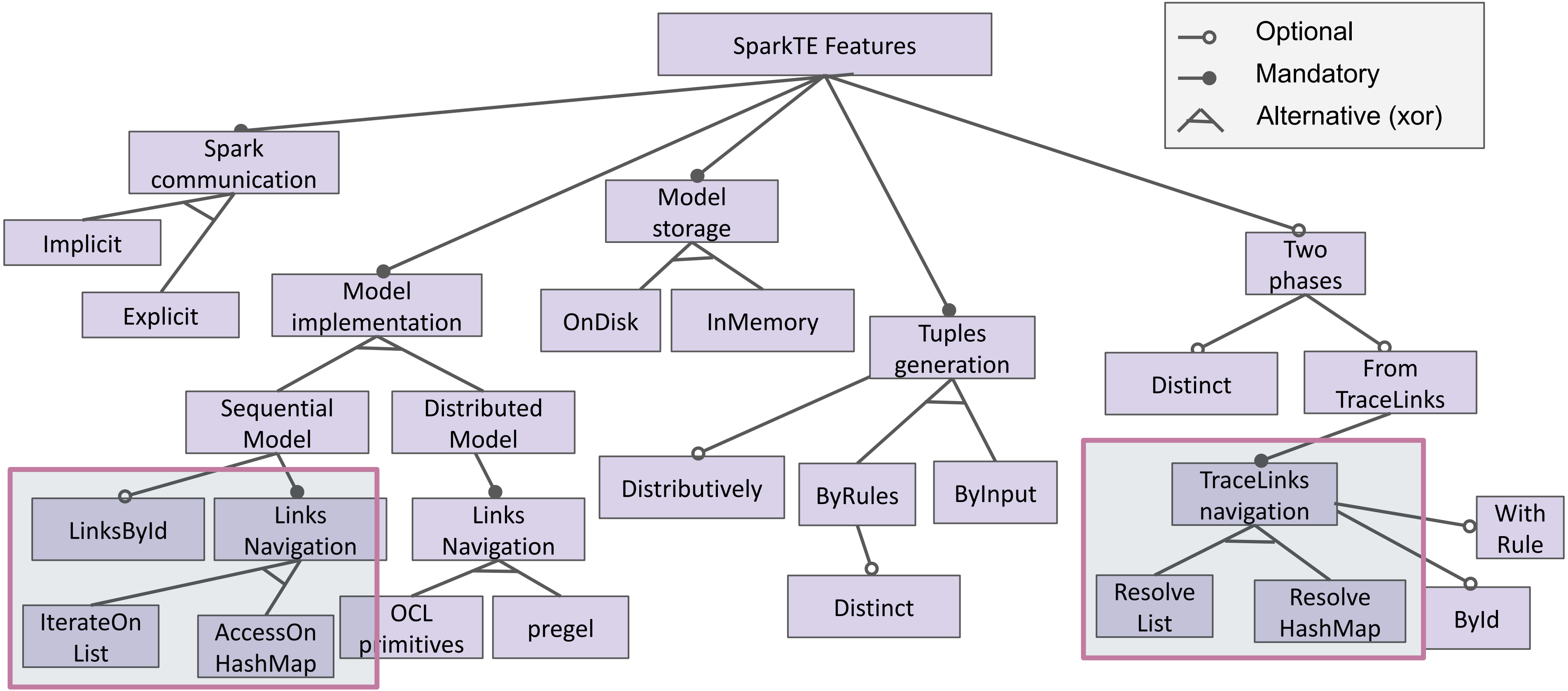
SparkTE feature diagram



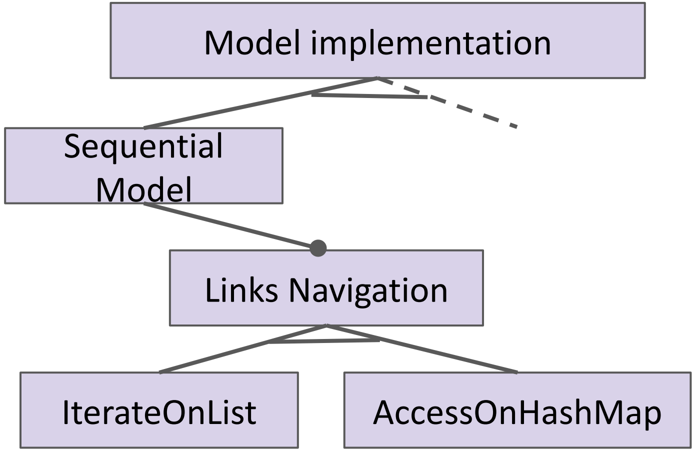
SparkTE feature diagram



SparkTE feature diagram

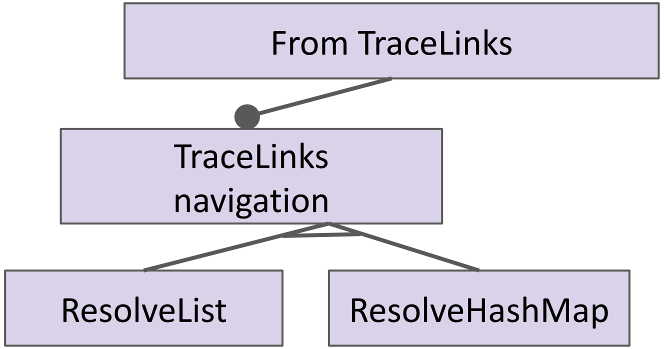


Feature 1: Link navigation strategy in sequential model



- IterateOnList :
  - Navigation by iteration
  - Simple to set-up
- AccessOnHashMap :
  - Additional computation in model loading
  - Increase memory usage
  - Direct access on links from elements

Feature 2: Trace-Links structure and resolution in apply phase



- ResolveList :
  - Resolution by iteration
  - Naturally gathered by master node
- ResolveHashMap :
  - Additional computation in instantiate phase
  - Increase memory usage
  - Fastest resolution

Using configurable engine to find features synergie

Execution of **Identity** transformation on a model of 100k elements and 250k links (4 cores)

Configuration 1: Links navigation	Configuration 2: TraceLinks navigation	Computation time (sec)	Instantiate phase (sec)	Apply phase (sec)
IterateOnList	ResolveList	1636 sec	3 sec	1633 sec
IterateOnList	ResolveHashMap	1584 sec	3 sec	1581 sec
AccessOnHashMap	ResolveList	233 sec	6 sec	227 sec
AccessOnHashMap	ResolveHashMap	12 sec	6 sec	6 sec

- **TraceLinks navigation**'s impact
  - on the **whole** computation is **negligible**
  - is **important** when **Links navigation** is processed by AccessOnHashMap
- **Links navigation**'s impact
  - decreases the **whole** computation time
  - increases the computation time of the instantiate phase

Design-space exploration for the **Find affinity** case

Feature label	Parallelizable CoqTL design choices (C1)	Optimal design choices (C2)
Model implementation	Sequential Model	Sequential Model
○ linksById	false	false
● **Link Navigation**	**IterateOnList**	**ResolveHashMap**
Model storage	InMemory	InMemory
Spark communication	**Implicit**	**Explicit**
Tuples generation	**ByRules**	**ByInput**
○ Distributively	false	false
○ **Distinct**	**false**	**true**
TraceLinks Navigation	ResolveList	ResolveList
○ byId	false	false
○ **withRule**	**false**	**true**
○ **Distinct**	**false**	**true**

Design-space exploration for the Find affinity case

Feature label	Parallelizable CoqTL design choices (C1)	Optimal design choices (C2)
Model implementation	Sequential Model	Sequential Model
○ linksById	false	false
● **Link Navigation**	**IterateOnList**	**ResolveHashMap**
Model storage	InMemory	InMemory
Spark communication	**Implicit**	**Explicit**
Tuples generation	**ByRules**	**ByInput**
○ Distributively	false	false
○ **Distinct**	**false**	**true**
TraceLinks Navigation	ResolveList	ResolveList
○ byId	false	false
○ **withRule**	**false**	**true**
○ **Distinct**	**false**	**true**

#elements	#links	C1 computation time	C2 computation time
1000	3000	9.799 sec	4.978 sec
2500	7300	81.047 sec	7.803 sec
5000	15000	882.708 sec	19.127 sec
7500	22000	> 2h	36.928 sec
10000	45000	Timeout error	65.198 sec

- ➢ The feature model is useful for comparing implementations
- ➢ Gives useful insights about the engine
- ➢ Highlighted correlation between features

CONCLUSION

Contribution of the thesis

Problem 1:
Many solutions for executing rules distributively

Built a **distributed solution** from a **specification**

- Re-designed specification to make it distributable
- Made a proof of equivalence for optimizations
- Shown our solution is scalable

Problem 2:
Many solutions for executing queries distributively

Evaluated distributed execution **strategies for a query**

- Implemented three design-choices
- Proposed hybrid solution
- Performance variation depending on the strategy

Problem 3:
Need an unified proposition for comparing design choices

Formalized features in our distributed solution

- Shown the synergies between them
- Shown the impact on performance

Publications

- Jolan Philippe, Hélène Coullon, Massimo Tisi, Gerson Sunyé. **Towards Transparent Combination of Model Management Execution Strategies for Low-Code Development Platforms.** *23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS): Companion Proceedings*, Oct 2020, Montreal (Virtually), Canada. 10.1145/3417990.3420206. Hal-02952952

- Jolan Philippe, Massimo Tisi, Hélène Coullon, Gerson Sunyé. **Executing Certified Model Transformations on Apache Spark.** *14th ACM SIGPLAN International Conference on Software Language Engineering (SLE)*, Oct 2021, Chicago IL, United States. 10.1145/3486608.3486901. Hal-03343942

- <u>Ongoing:</u> Jolan Philippe, Massimo Tisi, Gerson Sunyé. **Analysis of the Design-Space of a Distributed Transformation Engine.** *Software and Systems Modeling (SoSyM)*

- *Several public Lowcomote deliverables*
  - **Concepts for Multi-paradigm distributed transformation**
  - **Scalable low-code artefact persistence and query**
  - **Multi-paradigm distributed transformation engine**

Future work

- **Automated design-space exploration for a given scenario**
  - A model of the input (e.g., topological metrics)
  - A model of the platform **(Spark and ≠)**
  - Constraints and requirements

- **Other parameters to optimize (≠ CPU time)**
  - Network bandwidth
  - Memory consumption
  - Energy consumption/production

\+ **Other execution strategies (≠ data-dist)**
  - Take advantage of Spark for task-distribution
  - Combine incrementality and laziness to distribution

Contribution to the Analysis of the Design-Space of a Distributed Transformation Engine

Jolan PHILIPPE

PhD Defense, speciality: Computer Science