Executing Certified Model Transformations on Apache Spark

Jolan Philippe
IMT Atlantique, LS2N

Massimo Tisi
IMT Atlantique, LS2N

Hélène Coullon
IMT Atlantique, Inria, LS2N

Gerson Sunyé
Univ. of Nantes, LS2N

SLE 2021, 17th October

Context: Correctness on large model transformation

Model transformation

An automated way of modifying and creating models

Correctness

Formally proving the respect to a specification of a model transformation

Large-model challenges

- Horizontal and Vertical Scalability
- Memory management
- Computation time

Context: Correctness on large model transformation

Model transformation

An automated way of modifying and creating models

Correctness

Formally proving the respect to a specification of a model transformation

Formalism and Software

Curry-Howard Correspondence
Proof-assistants

Large-model challenges

- Horizontal and Vertical Scalability
- Memory management
- Computation time

Hardware Solution

Multi-core, and distributed architecture

Software Solution

Framework for large-data management

Context: Correctness on large model transformation

Model transformation

An automated way of modifying and creating models

Correctness

Formally proving the respect to a specification of a model transformation

Formalism and Software

Curry-Howard Correspondence
Proof-assistants

Large-model challenges

- Horizontal and Vertical Scalability
- Memory management
- Computation time

Hardware Solution

Multi-core, and distributed architecture

Software Solution

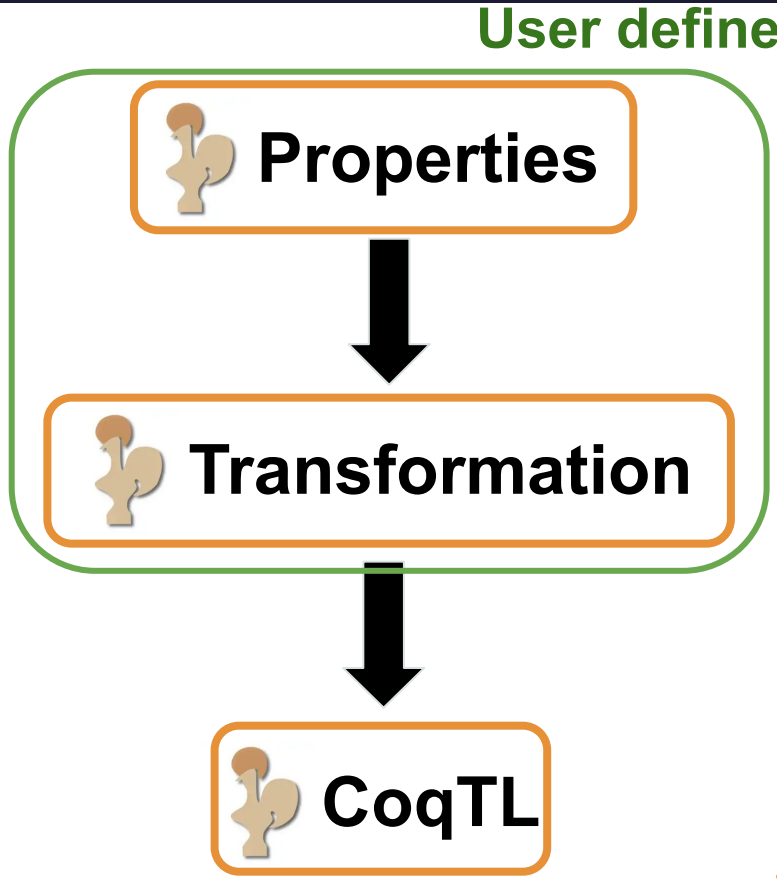
Framework for large-data management

No bridge (in a MDE context)

Background: CoqTL

- A DSL for expressing model transformations with Coq
- Allow to express user properties
- Proving mechanism

Reference: Zheng Cheng, Massimo Tisi, Rémi Douence. CoqTL: A Coq DSL for Rule-Based Model Transformation. *Software and Systems Modeling*, Springer Verlag, In press, pp.1-15.



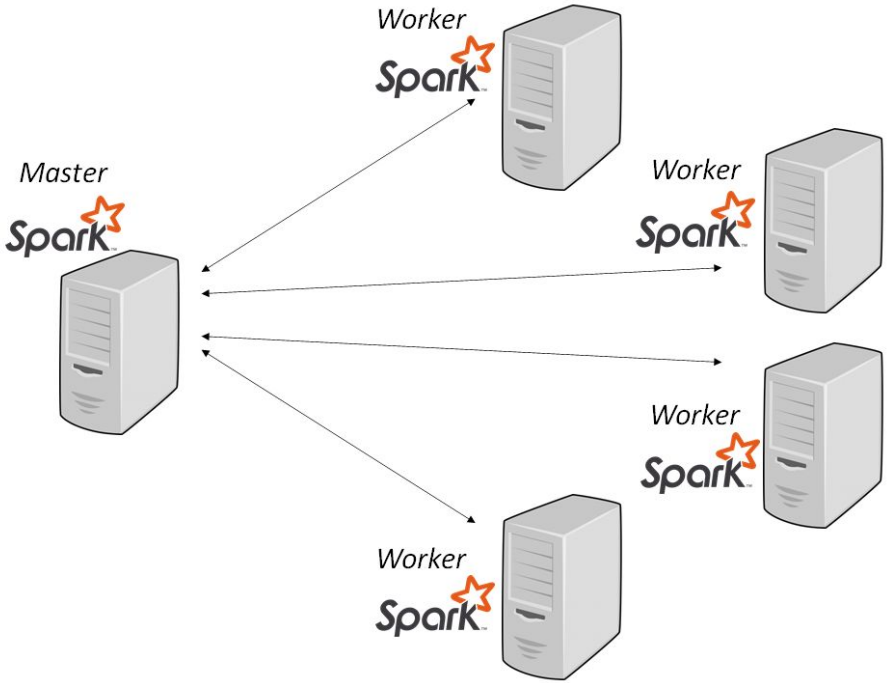
Background: Apache Spark

A unified analytics engine for large-scale data processing

- **Master** & **Workers** architecture
- In-memory computation

- High-order functions
  - Ease parallelism
  - **Distributed** lists

- Widely-used by data analysts
  - Many examples
  - **High-confidence**

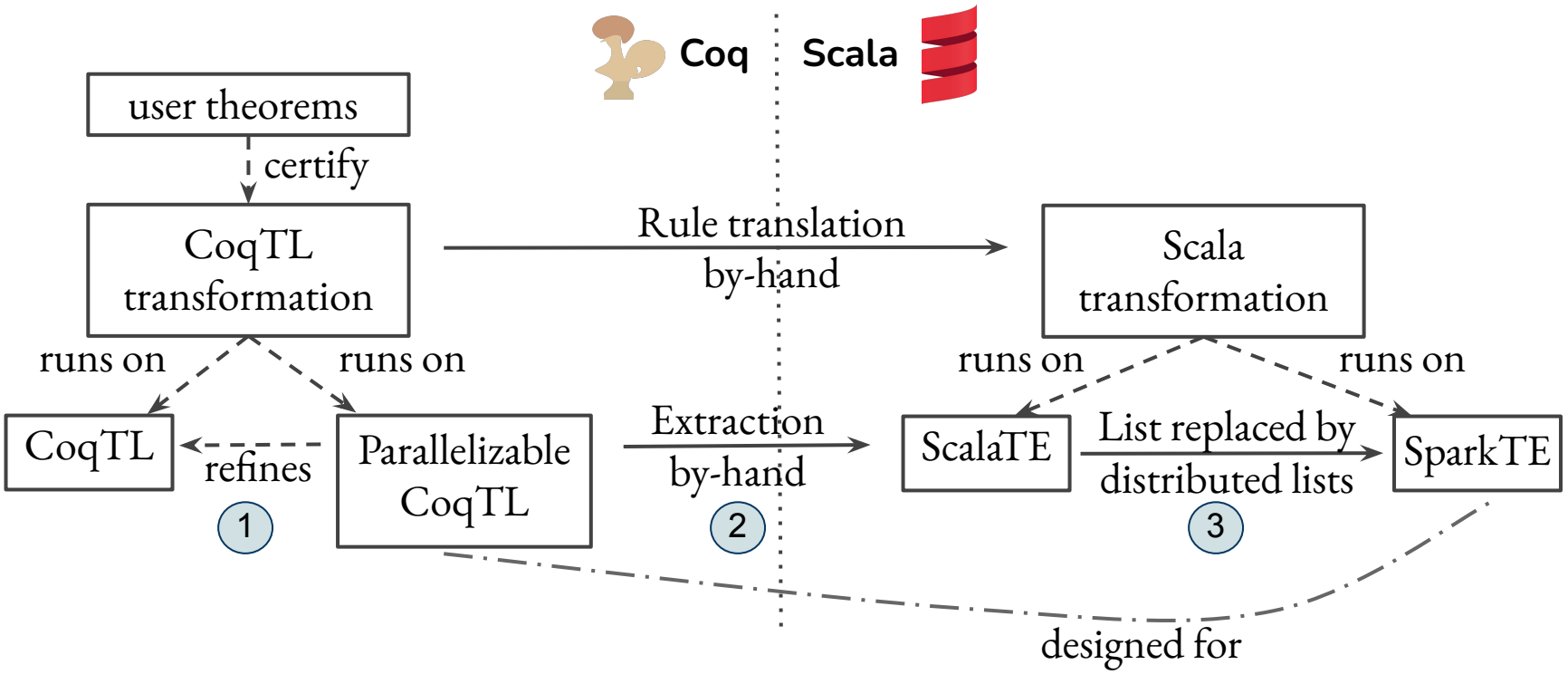


Objective

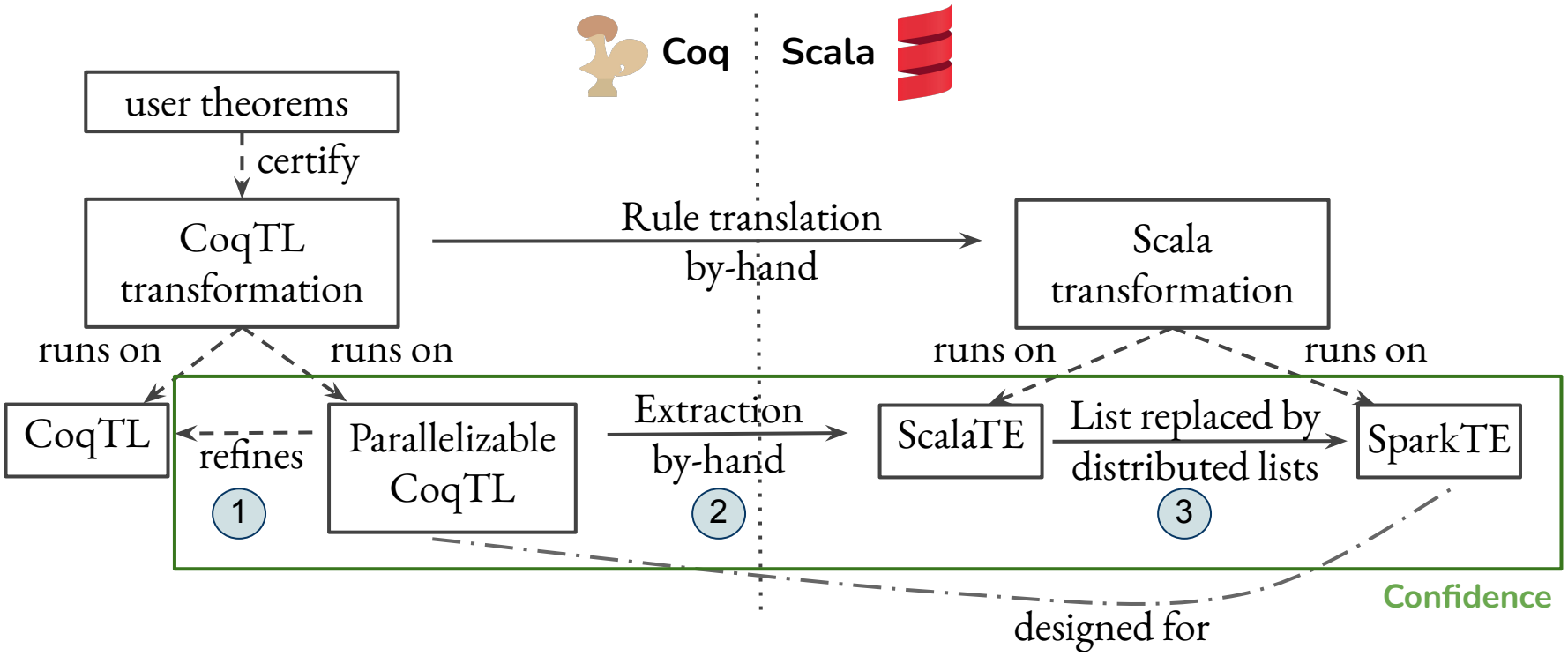
- Bridge correctness (**Coq**) to a distributed solution (**Spark**)
- Have **two** distinct **specifications**
  1. One designed for **reasoning** (CoqTL)
  2. One for the actual execution, **specifying the optimizations** (Parallelizable CoqTL)

  ➢ **Proof of equivalence** between those
- Have an executable solution on top of Spark: **SparkTE**

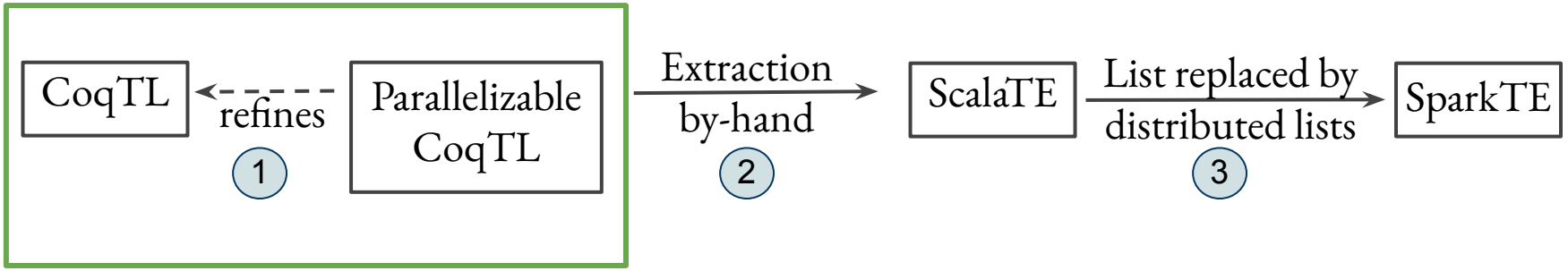
Coq to Scala


Coq
Scala
user theorems
certify
CoqTL transformation
Rule translation by-hand
Scala transformation
runs on
runs on
runs on
runs on
CoqTL
refines
1
Parallelizable CoqTL
Extraction by-hand
2
ScalaTE
List replaced by distributed lists
3
SparkTE
designed for

Coq to Scala

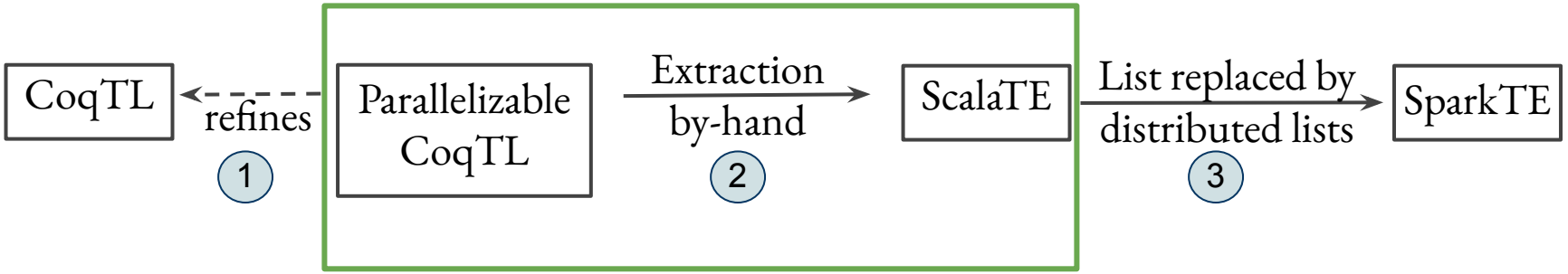

Coq
Scala
user theorems
certify
CoqTL transformation
Rule translation by-hand
Scala transformation
runs on
runs on
runs on
runs on
CoqTL
refines
1
Parallelizable CoqTL
Extraction by-hand
2
ScalaTE
List replaced by distributed lists
3
SparkTE
Confidence
designed for

Parallelizable CoqTL refines CoqTL



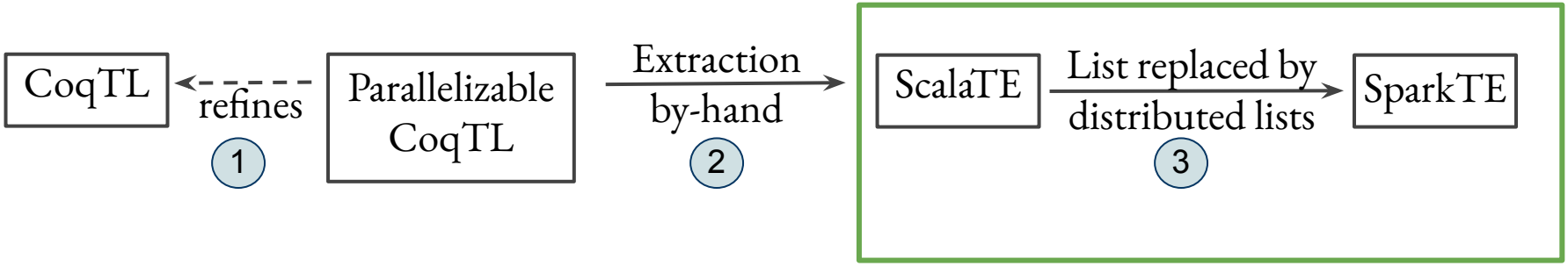
- A CoqTL refinement: Parallelizable CoqTL
  - Designed to increase parallelization
- **Confidence? Formal proof of equivalence with standard CoqTL**

Parallelizable CoqTL to ScalaTE



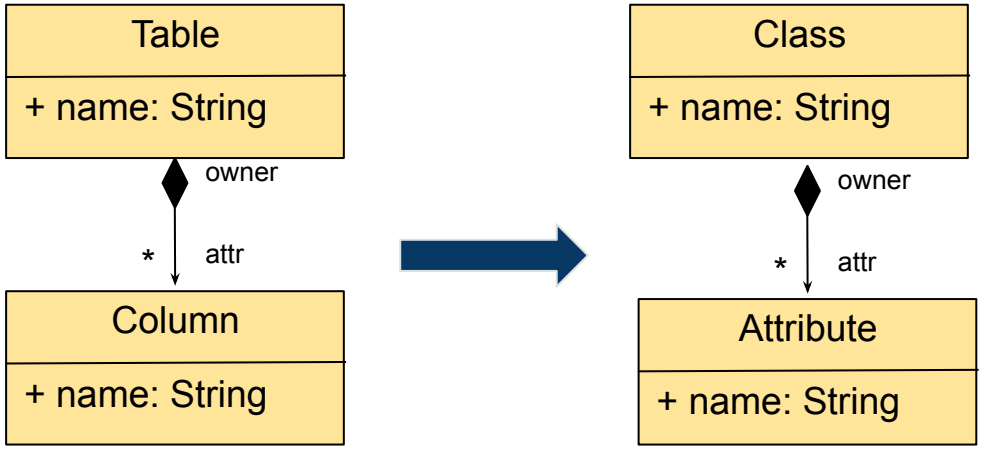
- An implementation written in Scala
  - Manually extracted
  - Executable solution as target: Scala Transformation Engine (ScalaTE)
- **Confidence ? Direct translation of pure functions**

Parallelizable CoqTL refines CoqTL



- A parallel implementation on top of Spark
  - Replace lists by distributed lists
  - Executable solution on distributed architectures as target: SparkTE
- **Confidence ? “In Spark we trust”**

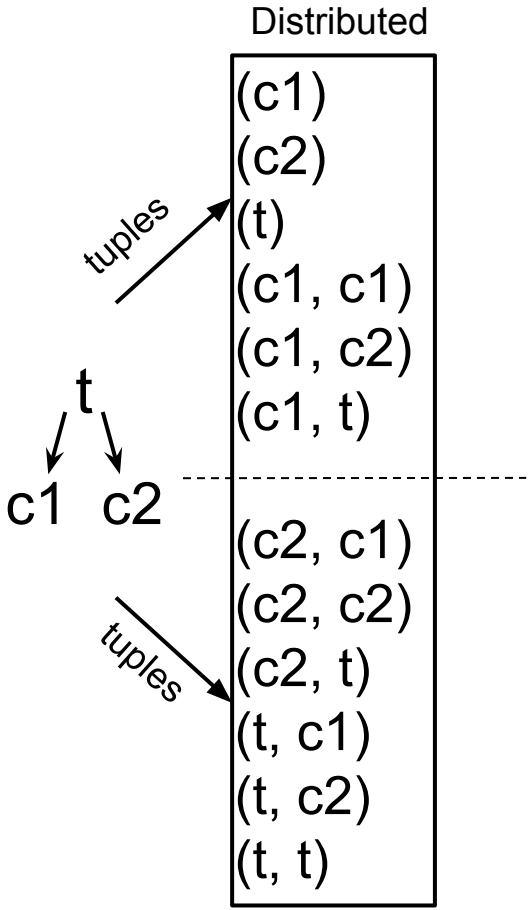
Simple example: Relational2Class



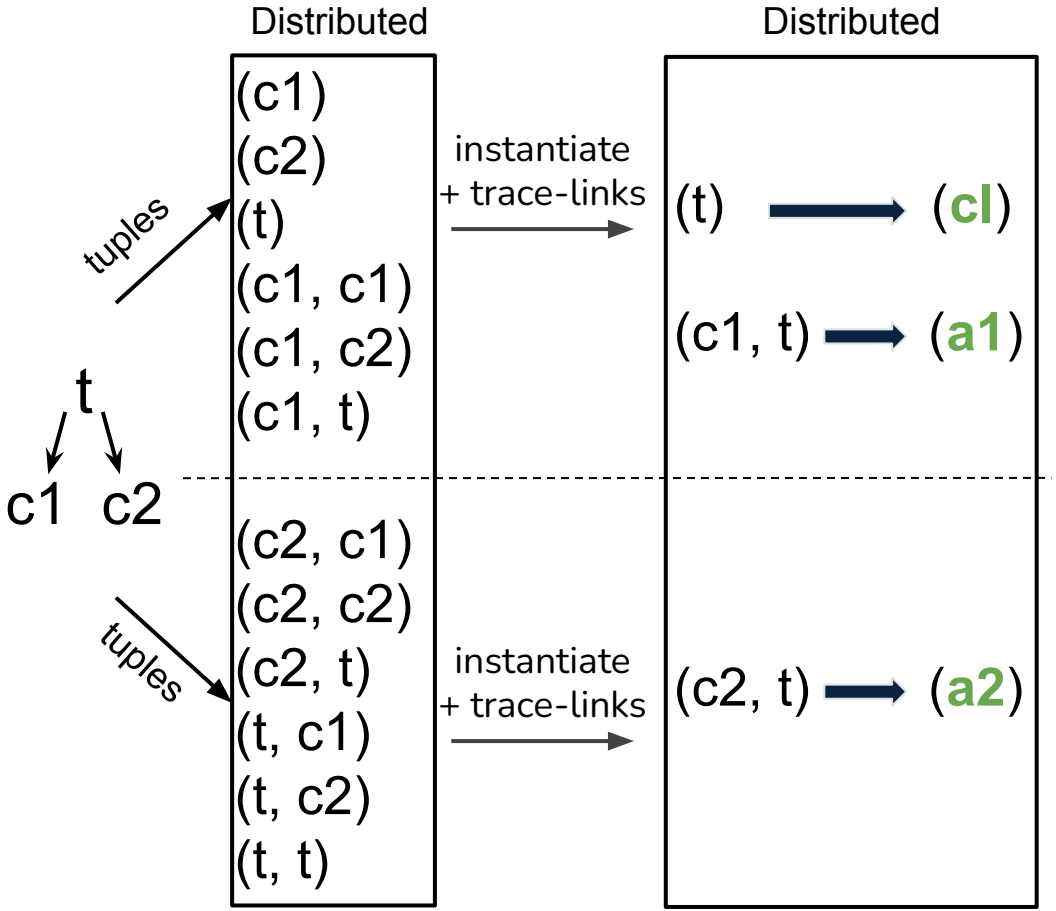

```
rule Table2Class {
        from s: RelationalTable
        to t: ClassClass (s.name)
            r: ClassToAttributes(t, resolveAll(s.attr))
}
```

```
rule Column2Attribute {
        from
            s1: RelationalColumn
            s2: RelationalTable | s1.owner == s2
        to t: ClassAttribute (s1.name)
            r: AttributeToClass(t, resolve(s2))
}
```

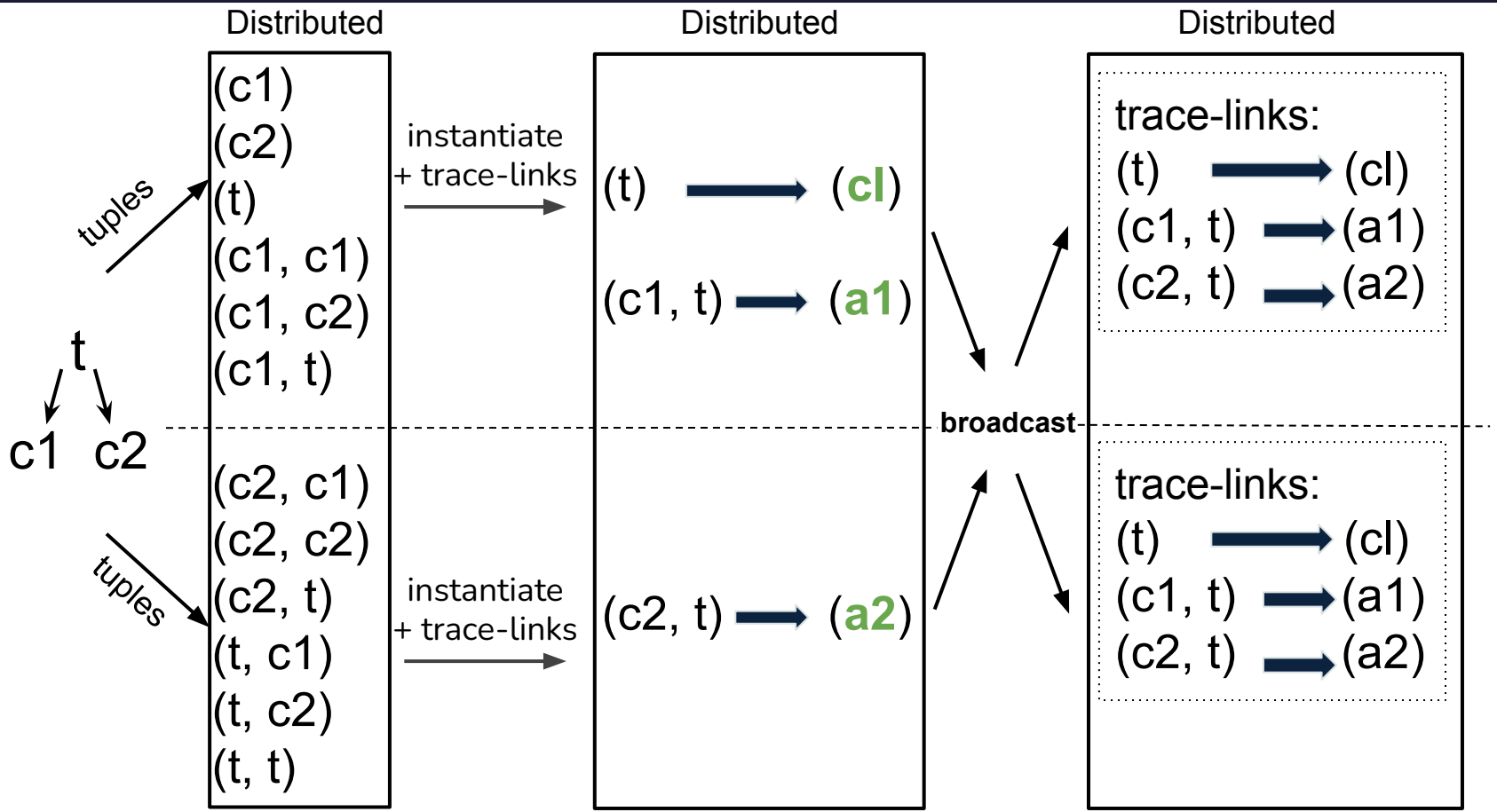
Running Relational2Class on SparkTE



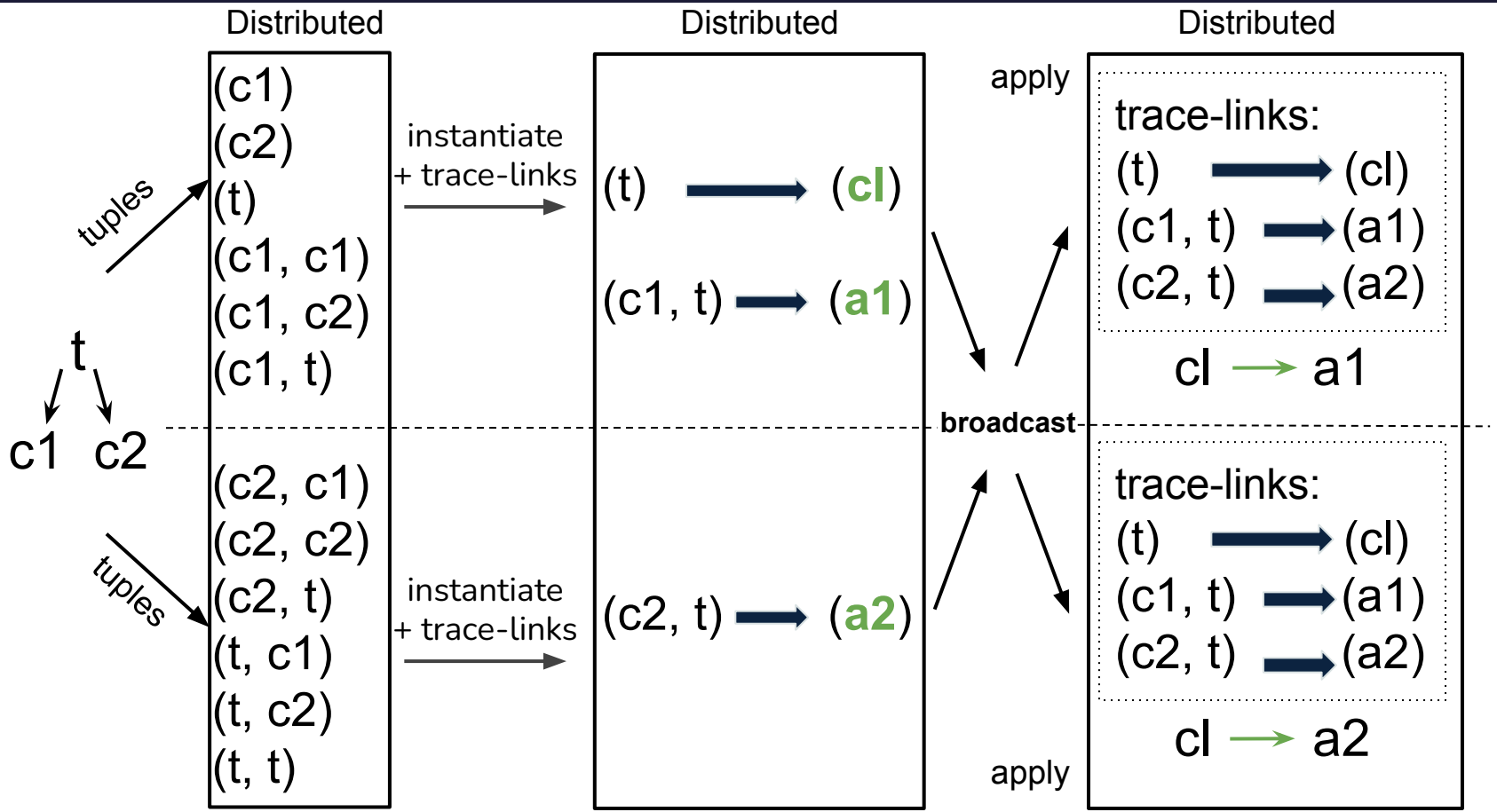
Running Relational2Class on SparkTE



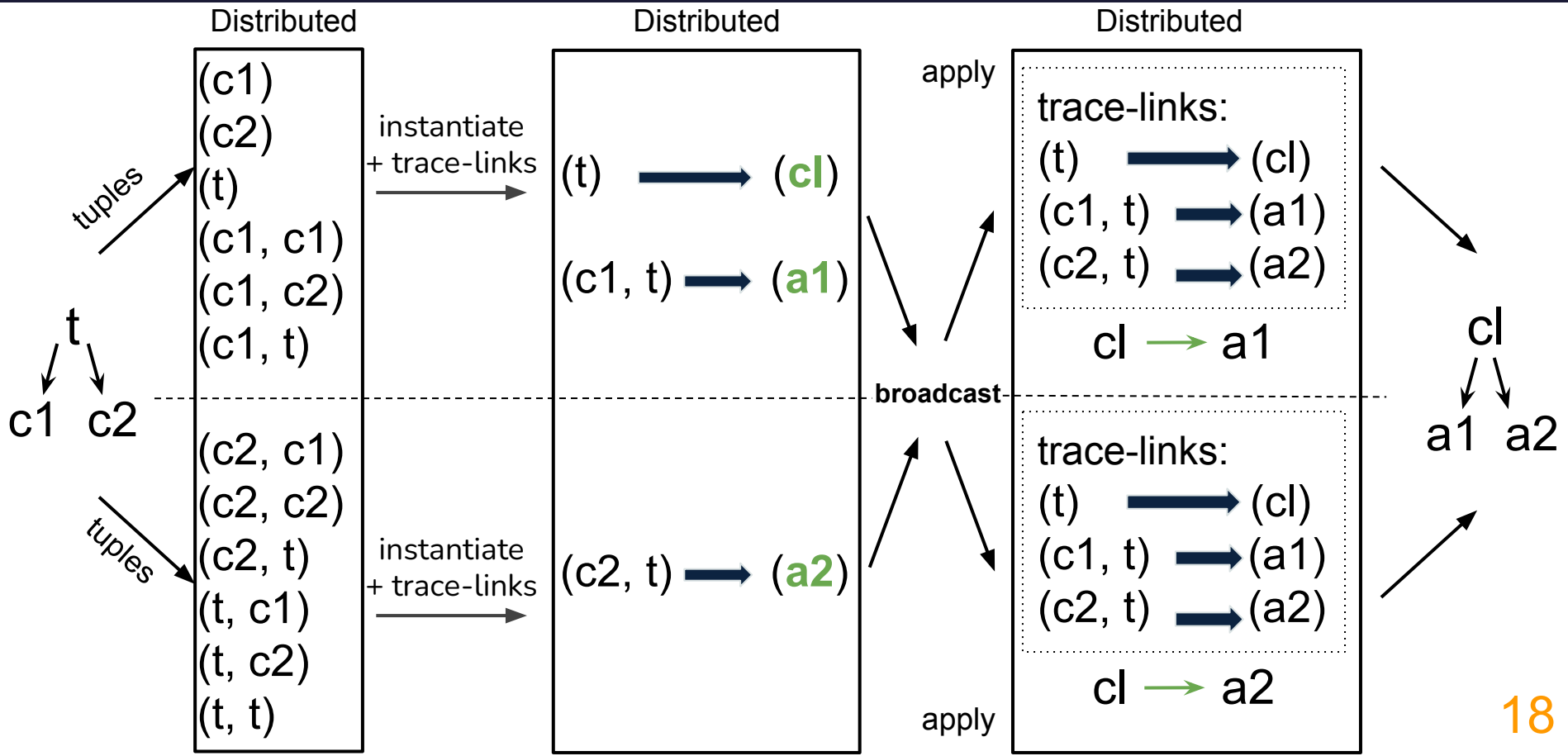
Running Relational2Class on SparkTE



Running Relational2Class on SparkTE



Running Relational2Class on SparkTE



Optimization 1: One-phase to two-phases approach

Standard CoqTL specification
- Single phase
- A rule generates output element and output link at the same time
- Easy for reasoning

- Two independent versions of CoqTL: Standard + Parallelizable
- A two-phases implementation:
  - first the output elements
  - then the output links

Optimization 2: Tuples by rule

Standard CoqTL solution:

- A recursive algorithm
- Distribution at the last iteration
- Too many useless tuples
- Leads to an imbalance in partitions

Solution:

- Generate only the useful tuples
- Instead of iterating on elements, we iterate on rules
- More balanced partitions

Optimization 3: Use of traces

Standard CoqTL specification

- Complete application of rules for creating links (including instantiate)

Solution

- Use the trace-links in the **apply** phase
- Iterate on trace-links
  - Find corresponding rule
  - Find all involved output element (from other rules)
    - Using resolve function
  - Creates the output link

Experimental setup

Goal:
- Study the parallelization (scalability) of SparkTE

Versions:
- Scala 2.12
- Spark 3.1.1 (Hadoop 2.7)

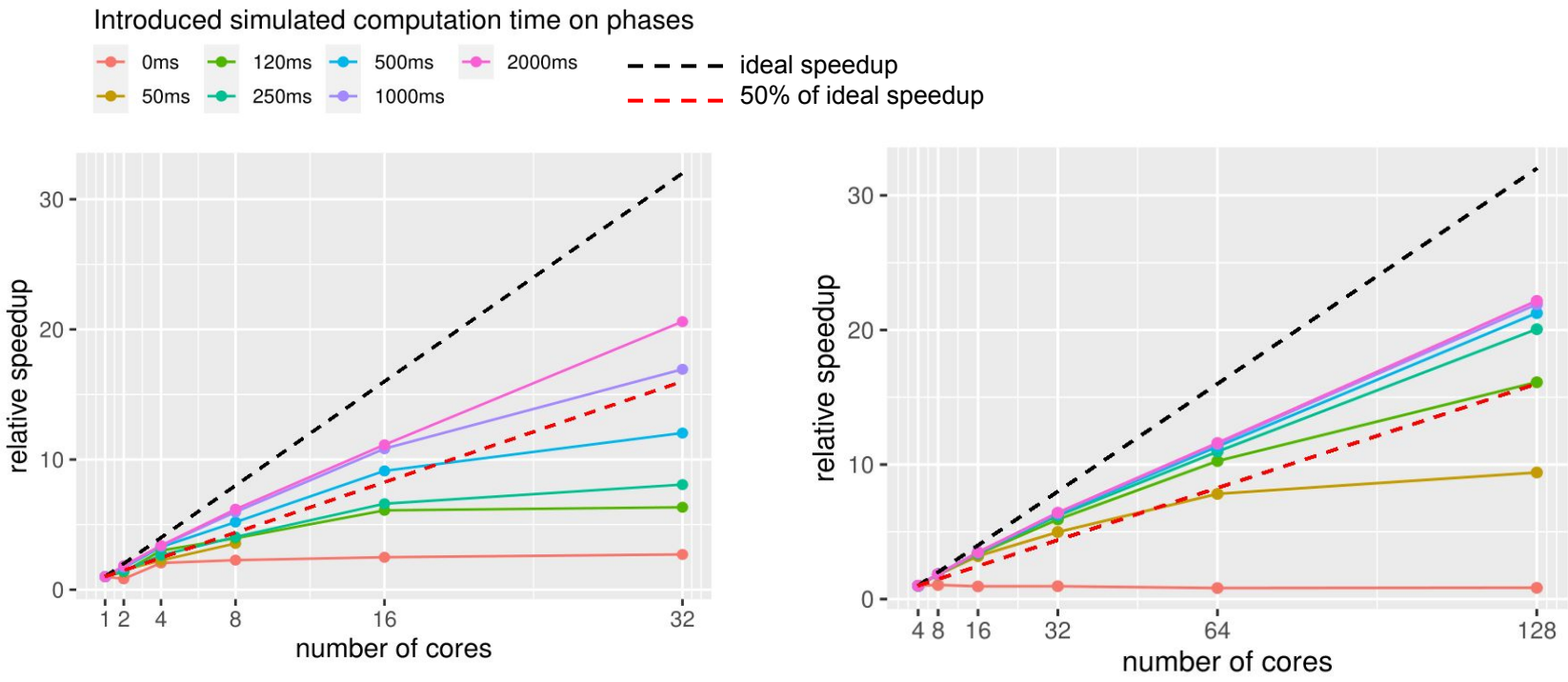
Cluster:
- Grid'5000 platform
  - *paravance* (Rennes): 2x8 cores/CPU, *Intel Xeon E5-2620* memory of 128GB
  - *gros* (Nancy): 2x18 cores/CPU, *Intel Xeon Gold 5220* memory of 96GB

Case studies:
- Relational2Class multivalued attributes
- IMDb "find couples" from TTC
- DBLP query

First experiment

	elements	links	complexity	Cores (machines)	1 (1)	2 (1)	4 (2)	8 (2)
Relational2Class	150	290	Low	times (s)	27.02	32.50	13.17	11.91
				speedup	1.00	0.84	2.13	2.31
DBLP	700	1886	Medium	times (s)	0.83	0.35	0.56	0.84
				speedup	1.00	2.37	1.49	0.99
IMDb	440	1968	High	times (s)	38.85	22.01	18.33	11.61
				speedup	1.00	1.74	2.09	3.30

Potential parallelization



Model of 150 elements and 290 links, up 32 cores on 4 machines (on *paravance*)

Model of 600 elements and 1060 links, up 128 cores on 8 machines (on *gros*)

Conclusion and future work

- **Contribution**
  - A CoqTL refinement: Parallelizable CoqTL
  - An executable counterpart, written in Scala
  - A parallel implementation of the executable part on top of Spark
- **Conclusive remarks**
  - We have shown a potential scalability for different operational cost
  - Many challenges (caused by memory issues, distribution, Spark overhead)
- **Future work**
  - Write a certified compiler from CoqTL to Scala
  - Study other approaches supported by Spark (e.g., GraphX)
  - Integrate the work with persistence solution (e.g., HDFS)

Conclusion and future work

- **Contribution**
  - A CoqTL refinement: Parallelizable CoqTL
  - An executable counterpart, written in Scala
  - A parallel implementation of the executable part on top of Spark
- **Conclusive remarks**
  - We have shown a potential scalability for different operational cost
  - Many challenges (caused by memory issues, distribution, Spark overhead)
- **Future work**
  - Write a certified compiler from CoqTL to Scala
  - Study other approaches supported by Spark (e.g., GraphX)
  - Integrate the work with persistence solution (e.g., HDFS)

Questions ?