Formal approaches to:

Coordination-free query evaluation and multi-query optimization in parallel and distributed systems

Bas Ketsman

Outline

CALM Formalization

CALM Revision 1

Conclusion

} Coordination-free evaluation

Parallel-Correctness

Transferability

Conclusion

} Multi-Query optimization

Introduction

Context: Declarative Networking, where Datalog based languages are used for parallel and distributed computing in clusters with disordered communication.

CALM-conjecture: No-coordination $\stackrel{?}{=}$ Monotonicity

[Hellerstein, 2010]

Introduction

Context: Declarative Networking, where Datalog based languages are used for parallel and distributed computing in clusters with disordered communication.

CALM-conjecture: No-coordination $\stackrel{?}{=}$ Monotonicity

[Hellerstein, 2010]

[Ameloot, Neven, Van den Bussche, 2011]: TRUE

Introduction

Context: Declarative Networking, where Datalog based languages are used for parallel and distributed computing in clusters with disordered communication.

CALM-conjecture: No-coordination $\stackrel{?}{=}$ Monotonicity

[Hellerstein, 2010]

[Ameloot, Neven, Van den Bussche, 2011]: TRUE

[Zinn, Green, Ludäscher, 2012]: FALSE

Introduction

Context: Declarative Networking, where Datalog based languages are used for parallel and distributed computing in clusters with disordered communication.

CALM-conjecture: No-coordination $\stackrel{?}{=}$ Monotonicity

[Hellerstein, 2010]

[Ameloot, Neven, Van den Bussche, 2011]: TRUE

- for a setting where nodes have **no** information about the horizontal-distribution of records

[Zinn, Green, Ludäscher, 2012]: FALSE

Introduction

Context: Declarative Networking, where Datalog based languages are used for parallel and distributed computing in clusters with disordered communication.

CALM-conjecture: No-coordination $\stackrel{?}{=}$ Monotonicity

[Hellerstein, 2010]

[Ameloot, Neven, Van den Bussche, 2011]: TRUE

- for a setting where nodes have **no** information about the horizontal-distribution of records

[Zinn, Green, Ludäscher, 2012]: FALSE

- for settings where nodes **have** information about the horizontal-distribution of record

Goal: To clarify the relation between **monotonicity** and **coordination** in asynchronous systems and to reveal the more complete picture

Outline

CALM Formalization

CALM Revision 1

Conclusion

} Coordination-free evaluation

Parallel-Correctness

Transferability

Conclusion

} Multi-Query optimization

Monotonicity

Definition

A query Q is monotone if $Q(\mathbf{I}) \subseteq Q(\mathbf{I} \cup \mathbf{J})$ for all database instances $\mathbf{I}$ and $\mathbf{J}$.

Notation: $\mathcal{M}$ = class of monotone queries

Monotonicity

Definition

A query Q is monotone if $Q(\mathbf{I}) \subseteq Q(\mathbf{I} \cup \mathbf{J})$ for all database instances $\mathbf{I}$ and $\mathbf{J}$.

Notation: $\mathcal{M}$ = class of monotone queries

Example

- Q_Δ: Select triangles in a graph $\in \mathcal{M}$
- $Q_<$: Select open triangles in a graph $\notin \mathcal{M}$

Monotonicity

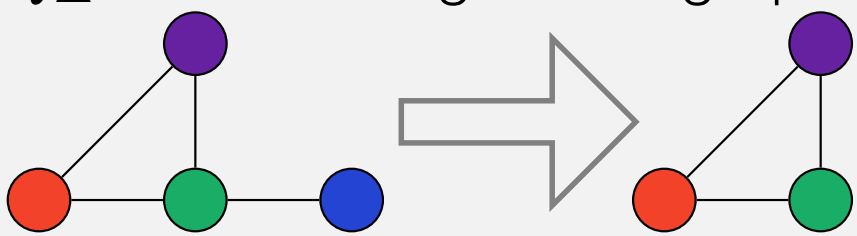
Definition

A query Q is monotone if $Q(\mathbf{I}) \subseteq Q(\mathbf{I} \cup \mathbf{J})$ for all database instances $\mathbf{I}$ and $\mathbf{J}$.

Notation: $\mathcal{M}$ = class of monotone queries

Example

- Q_Δ: Select triangles in a graph $\in \mathcal{M}$



- $Q_<$: Select open triangles in a graph $\notin \mathcal{M}$

Monotonicity

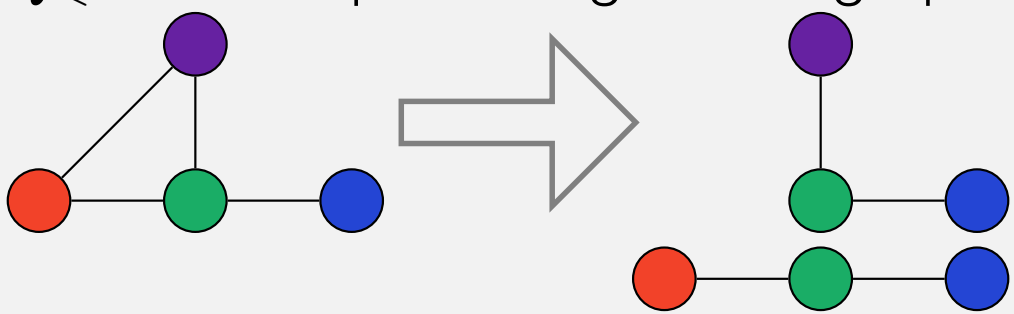
Definition

A query Q is monotone if $Q(\mathbf{I}) \subseteq Q(\mathbf{I} \cup \mathbf{J})$ for all database instances $\mathbf{I}$ and $\mathbf{J}$.

Notation: $\mathcal{M}$ = class of monotone queries

Example

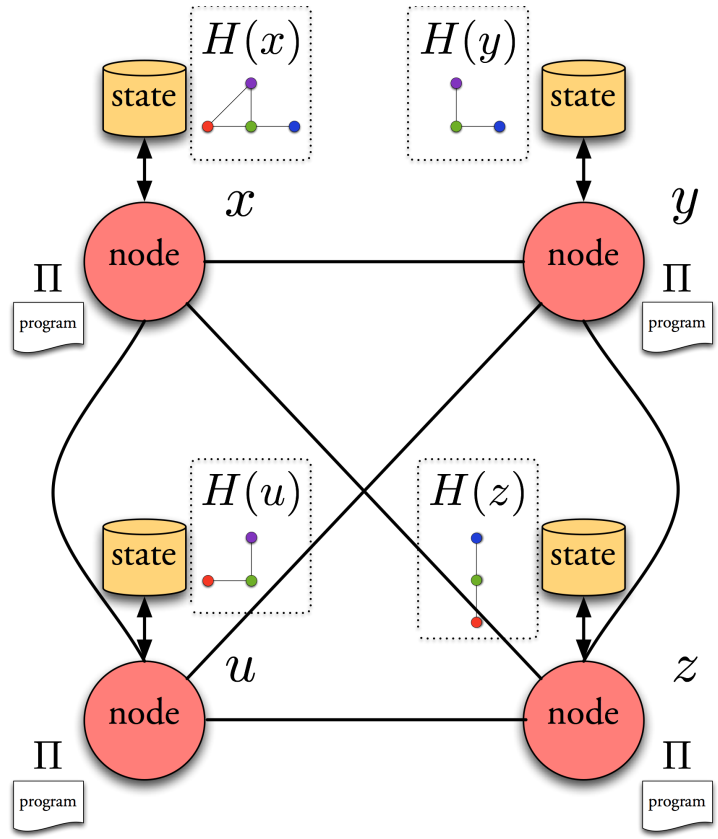
- Q_Δ: Select triangles in a graph $\in \mathcal{M}$
- $Q_<$: Select open triangles in a graph $\notin \mathcal{M}$



Relational Transducer Networks

[Ameloot, Neven, Van den Bussche, 2011]

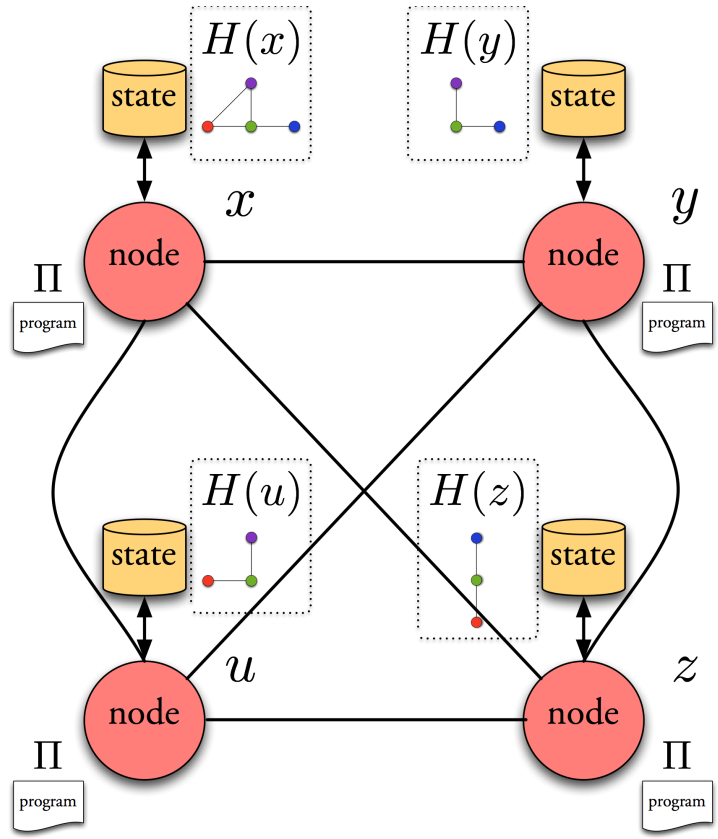
- Network $\mathcal{N} = \{x, y, u, z\}$
- Transducer Π
- messages can be arbitrarily delayed but never get lost



Relational Transducer Networks

[Ameloot, Neven, Van den Bussche, 2011]

- Network $\mathcal{N} = \{x, y, u, z\}$
- Transducer Π
- messages can be arbitrarily delayed but never get lost



Semantics defined in terms of runs over a transition system

Eventual Consistent Query Evaluation

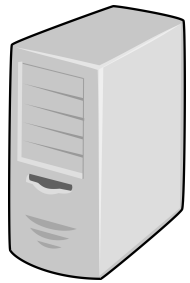
Definition

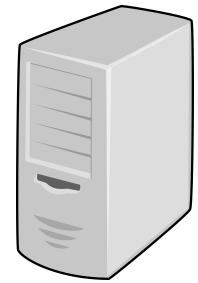
A transducer Π computes a query Q if

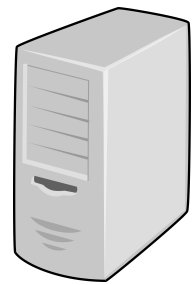
- for all networks $\mathcal{N}$, ← Network independent
- for all databases $\mathbf{I}$, ← Distribution independent
- for all horizontal distributions H, and
- for every run of Π, ← Consistency requirement

$$out(\Pi) = Q(\mathbf{I}).$$

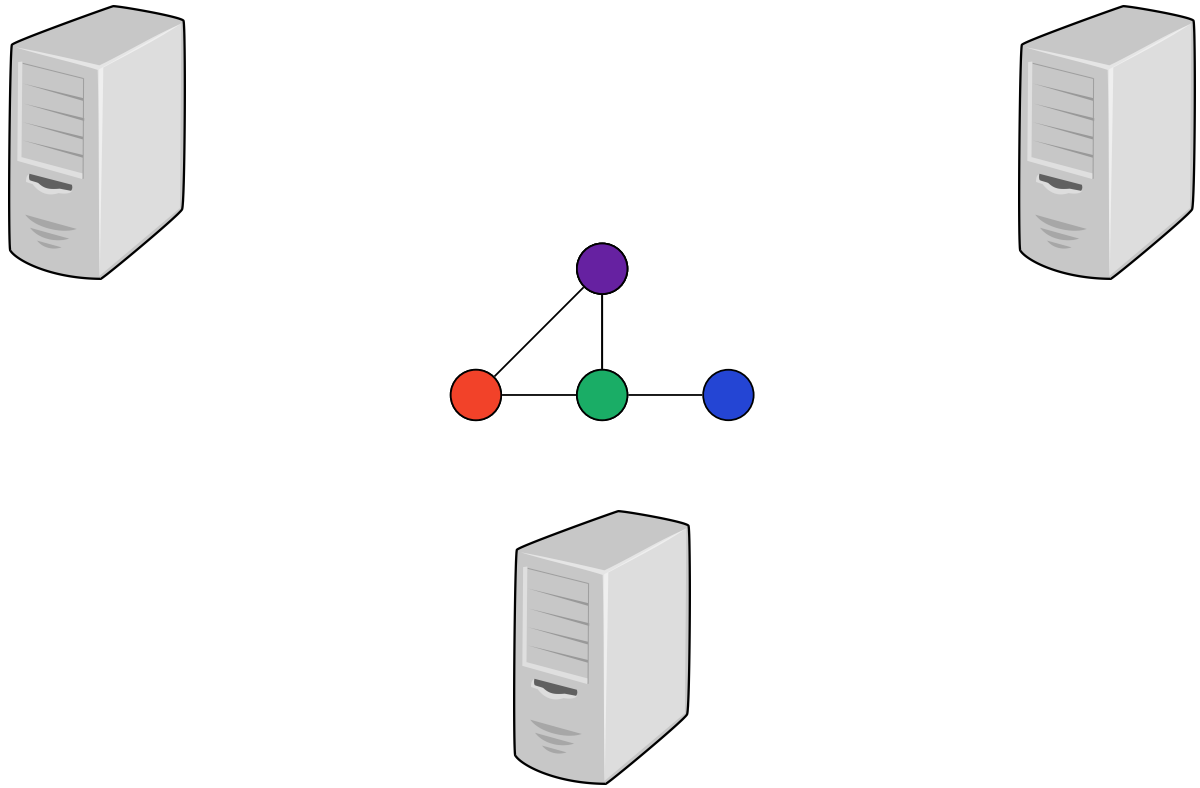
Example: $\mathcal{Q}_\Delta$: select all triangles



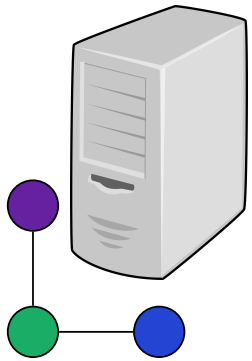


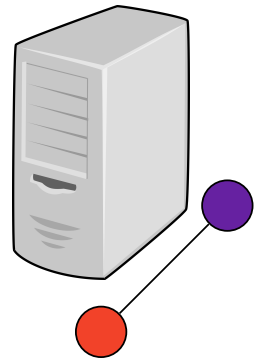


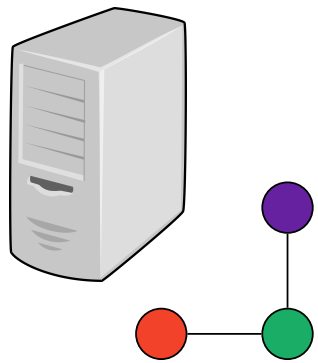
Example: $\mathcal{Q}_\Delta$: select all triangles



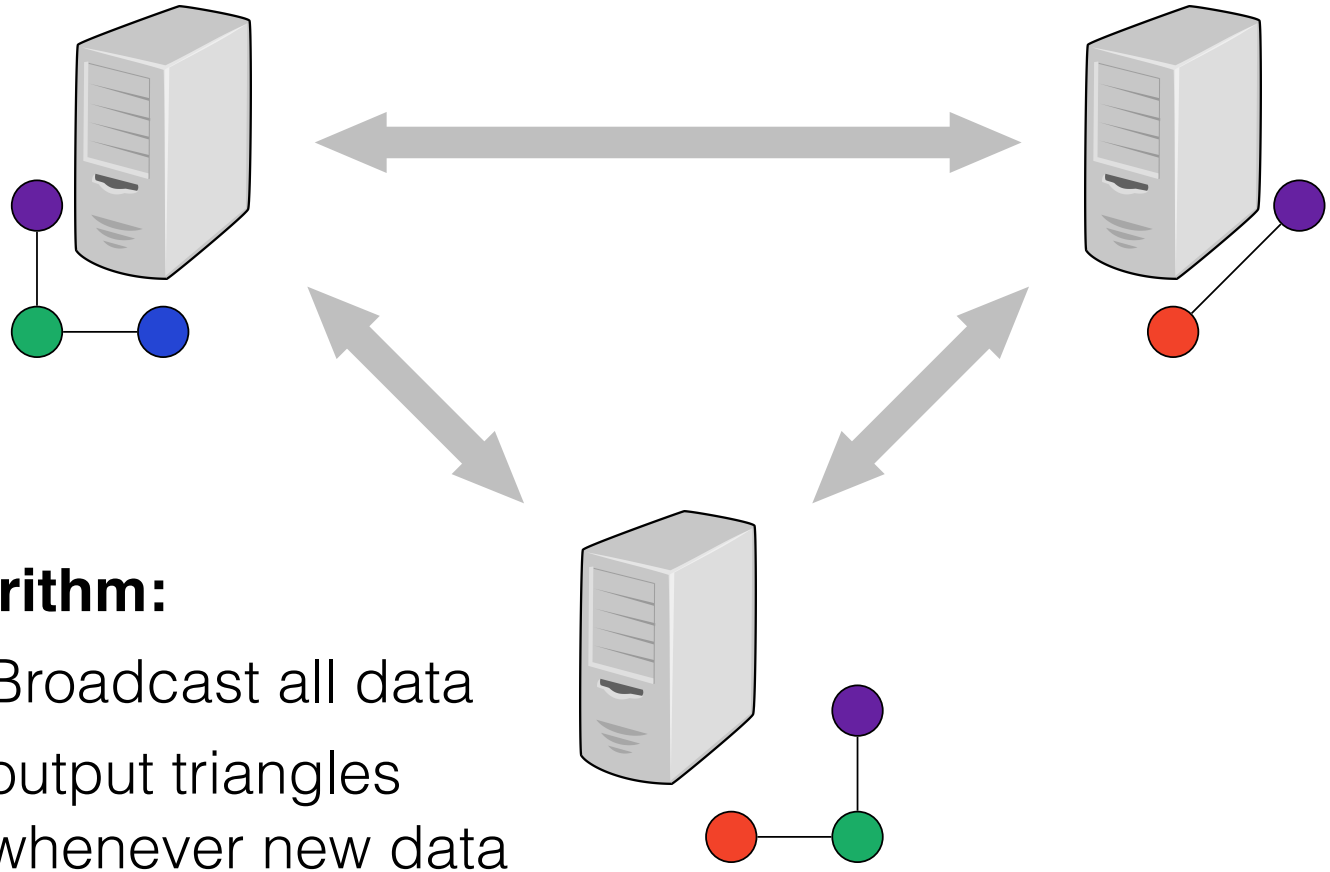
Example: $\mathcal{Q}_\Delta$: select all triangles







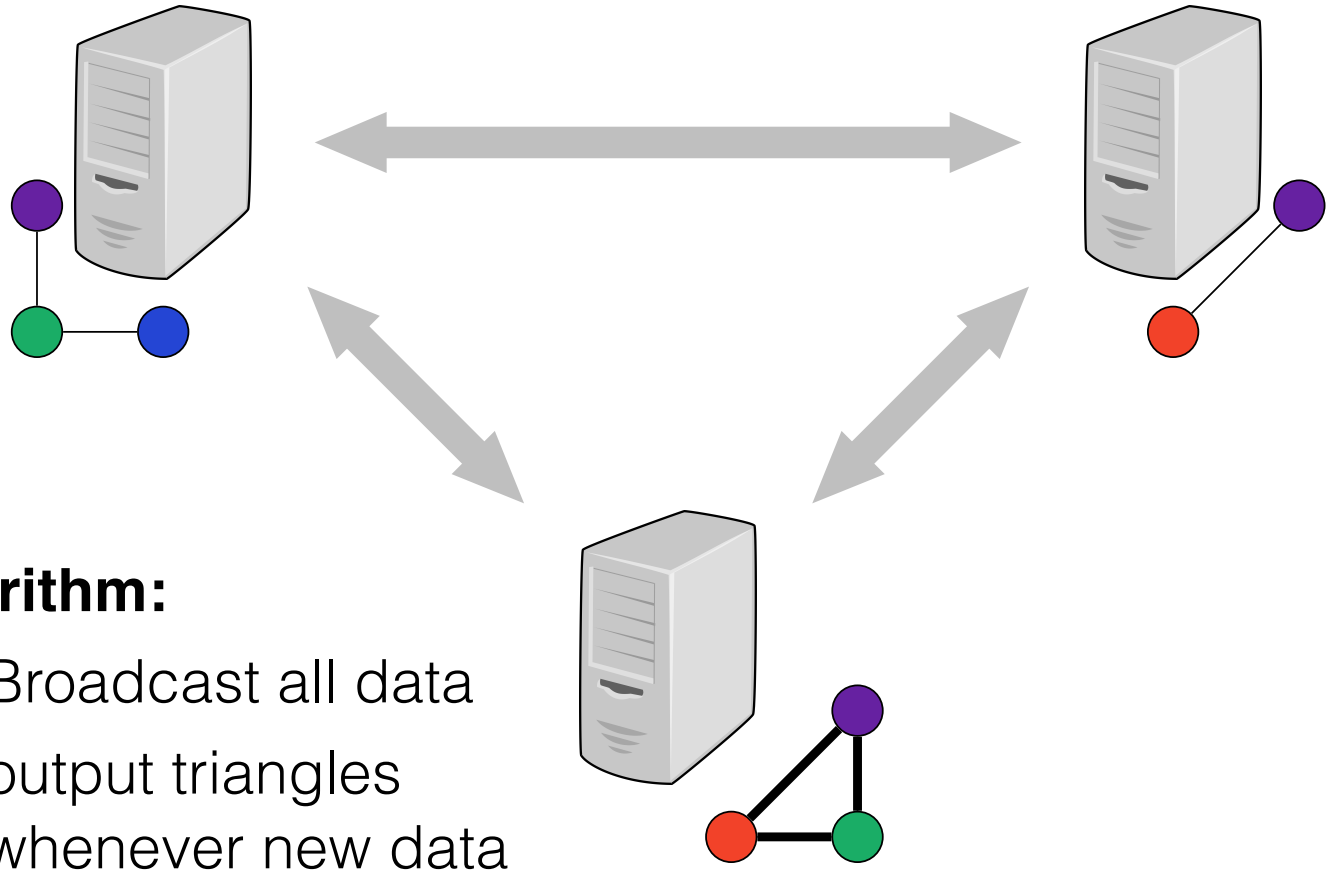
Example: $\mathcal{Q}_\Delta$: select all triangles



Algorithm:

- Broadcast all data
- output triangles whenever new data arrives

Example: $\mathcal{Q}_\Delta$: select all triangles

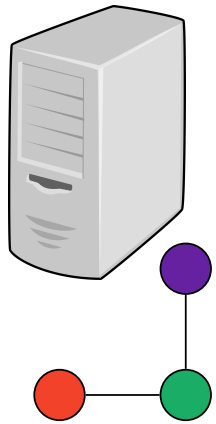


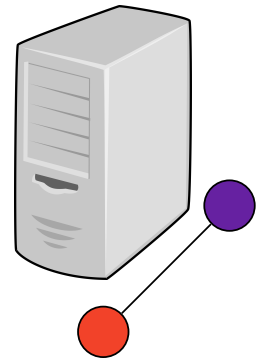
Algorithm:

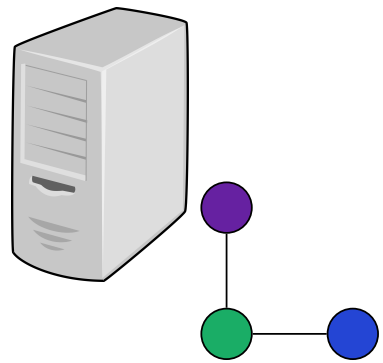
- Broadcast all data
- output triangles whenever new data arrives

Extremely naive, but works .. and is coordination-free!

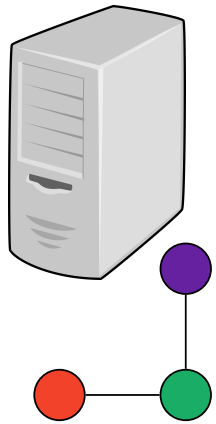
Example: $\mathcal{Q}_<$: select all open triangles

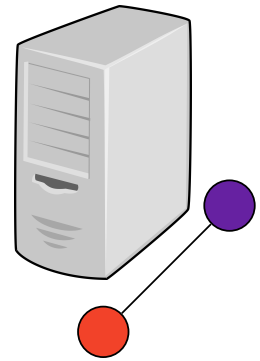


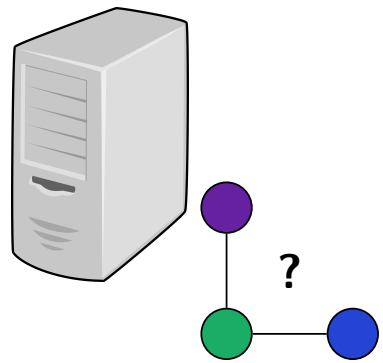




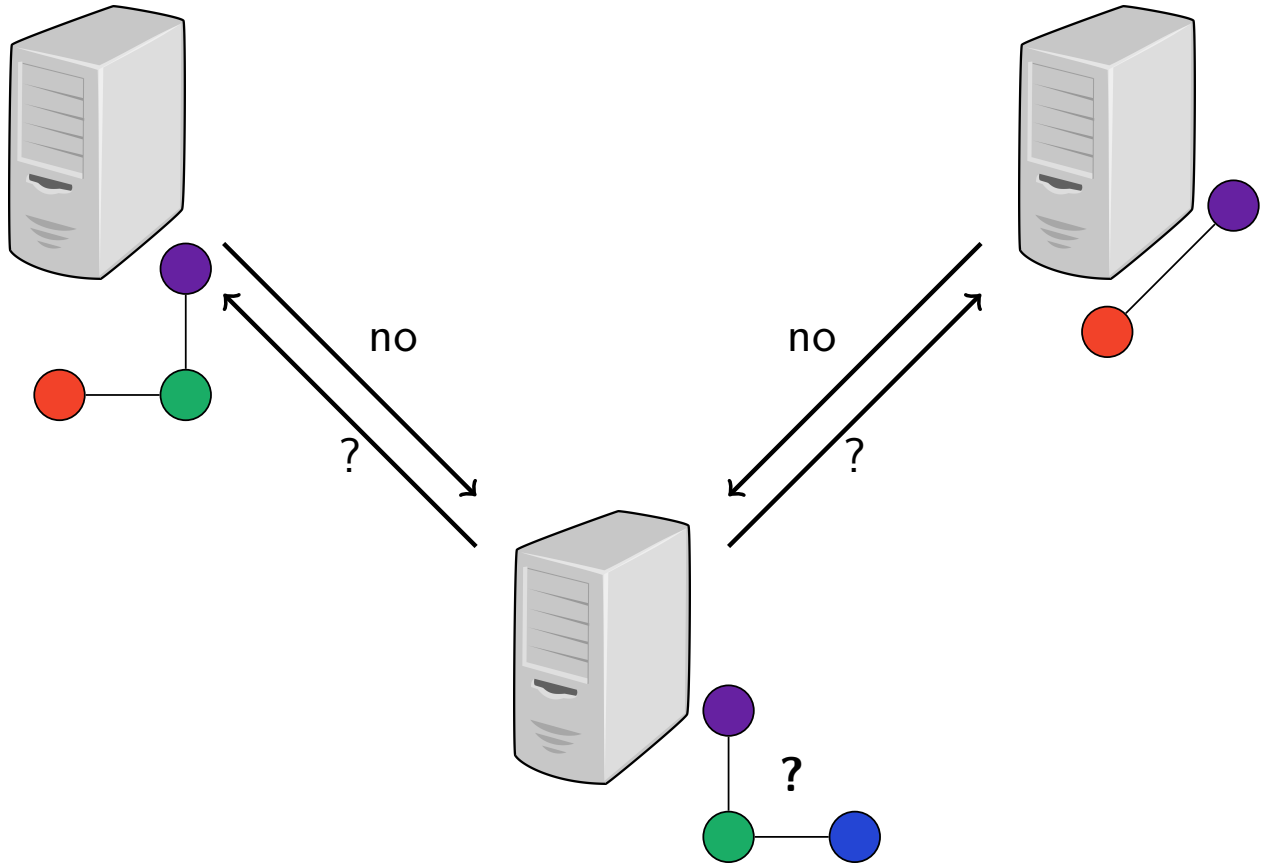
Example: $\mathcal{Q}_<$: select all open triangles



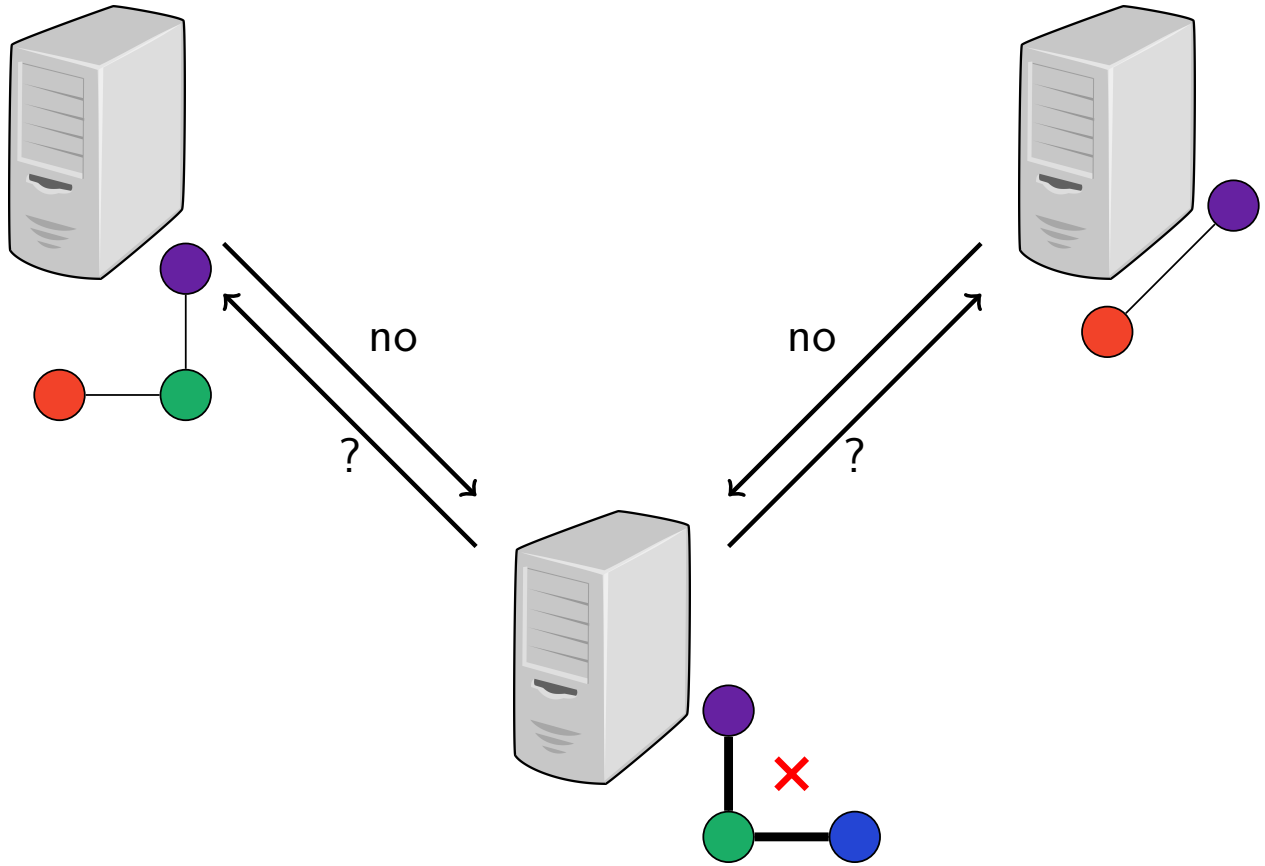




Example: $\mathcal{Q}_<$: select all open triangles



Example: $\mathcal{Q}_<$: select all open triangles



Coordination is needed to reason about the absence of records.

Coordination-freeness

Goal: separate data-communication from coordination-communication

Coordination-freeness

Goal: separate data-communication from coordination-communication

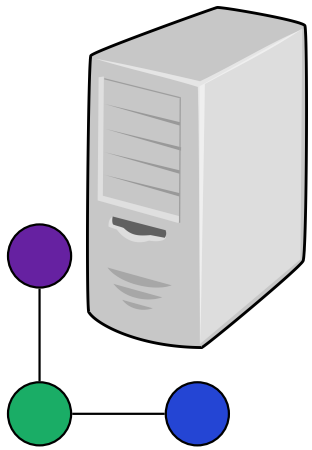
Definition

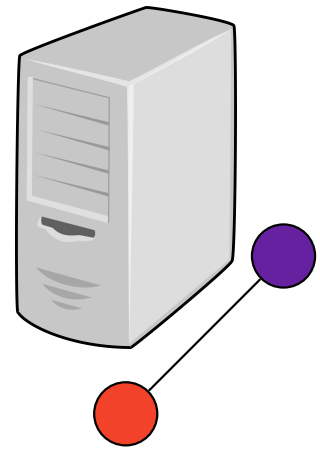
Π is coordination-free if for all inputs $\mathbf{I}$ there is a distribution on which Π computes $Q(\mathbf{I})$ without having to do communication.

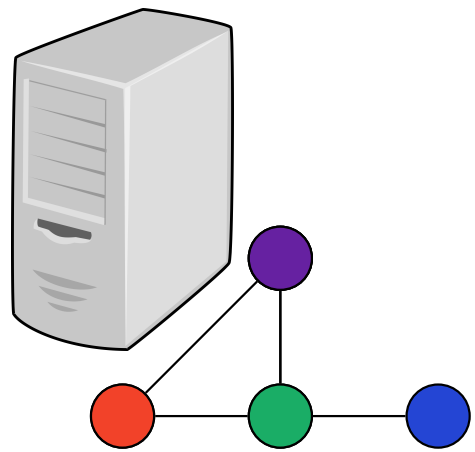
[Ameloot, Neven, Van den Bussche, 2011]

Example: Ideal Distribution

Q_Δ: select all triangles

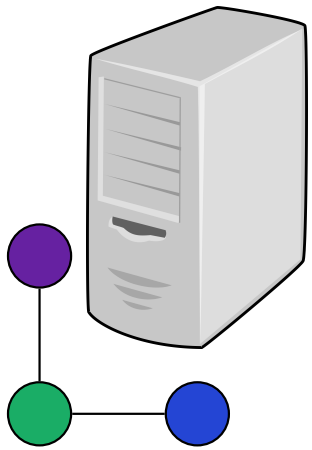


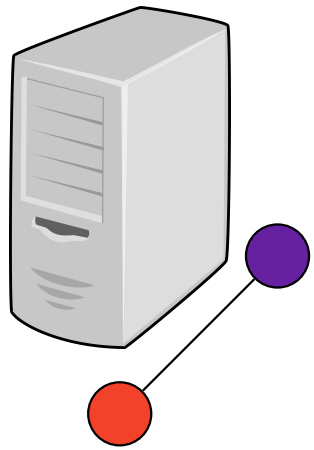




Example: Ideal Distribution

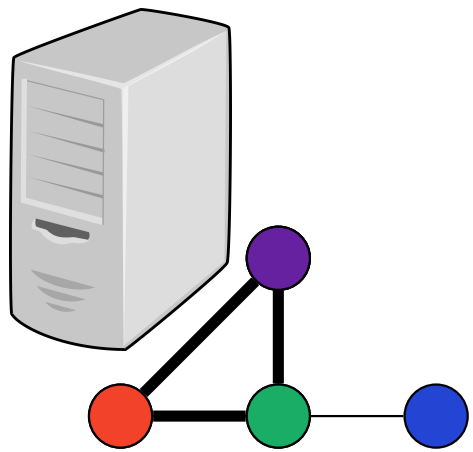
Q_Δ: select all triangles





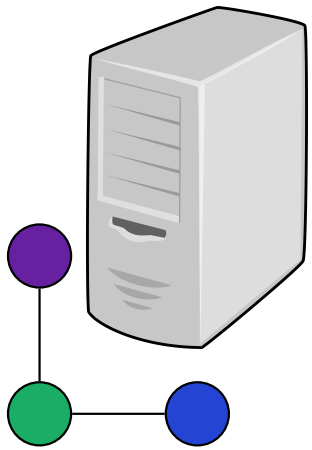
Algorithm:

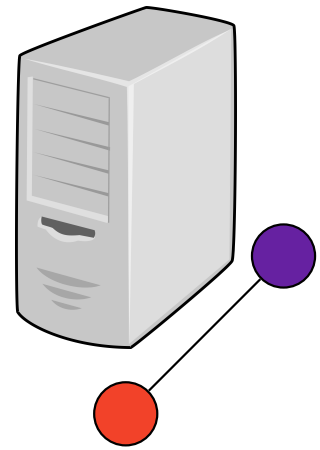
- Broadcast all data
- Output triangles whenever new data arrives



Example: Ideal Distribution

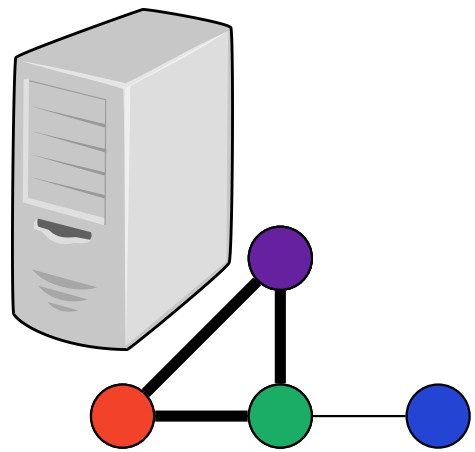
Q_Δ: select all triangles





Algorithm:

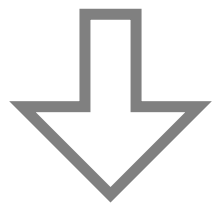
- (Broadcast all data)
- Output triangles whenever new data arrives



CALM-conjecture

[Ameloot, Neven, Van den Bussche, 2011]

A query has a coordination-free and eventually consistent execution strategy
iff
the query is monotone



Theorem

$\mathcal{F}_0 = \mathcal{M}$

Definition

$\mathcal{F}_0 =$ set of queries which are distributedly computed by coordination-free transducers

Outline

CALM Formalization

CALM Revision 1

Conclusion

} Coordination-free evaluation

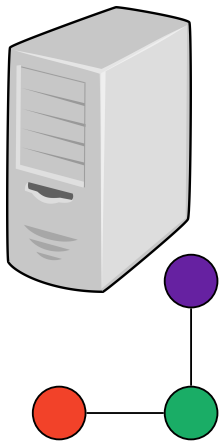
Parallel-Correctness

Transferability

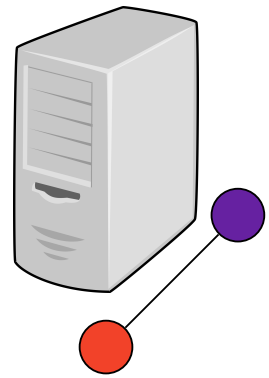
Conclusion

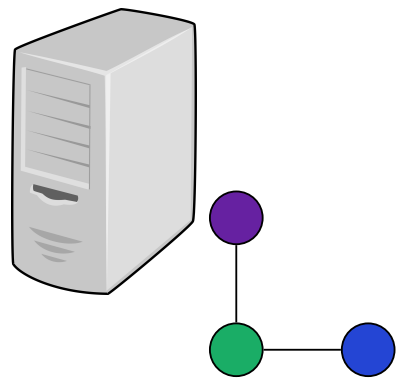
} Multi-Query optimization

Policy-aware Transducers

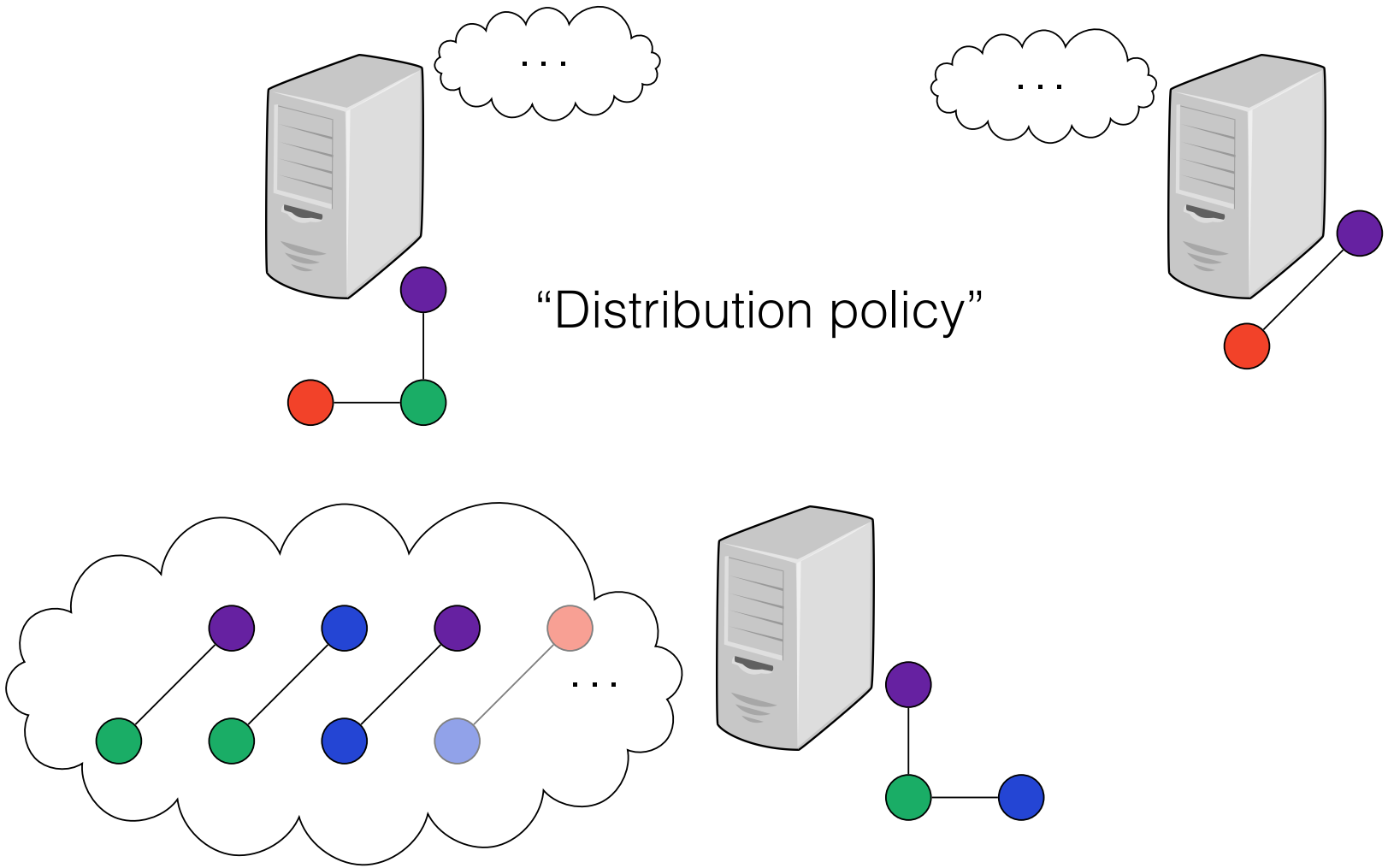


“Distribution policy”





Policy-aware Transducers

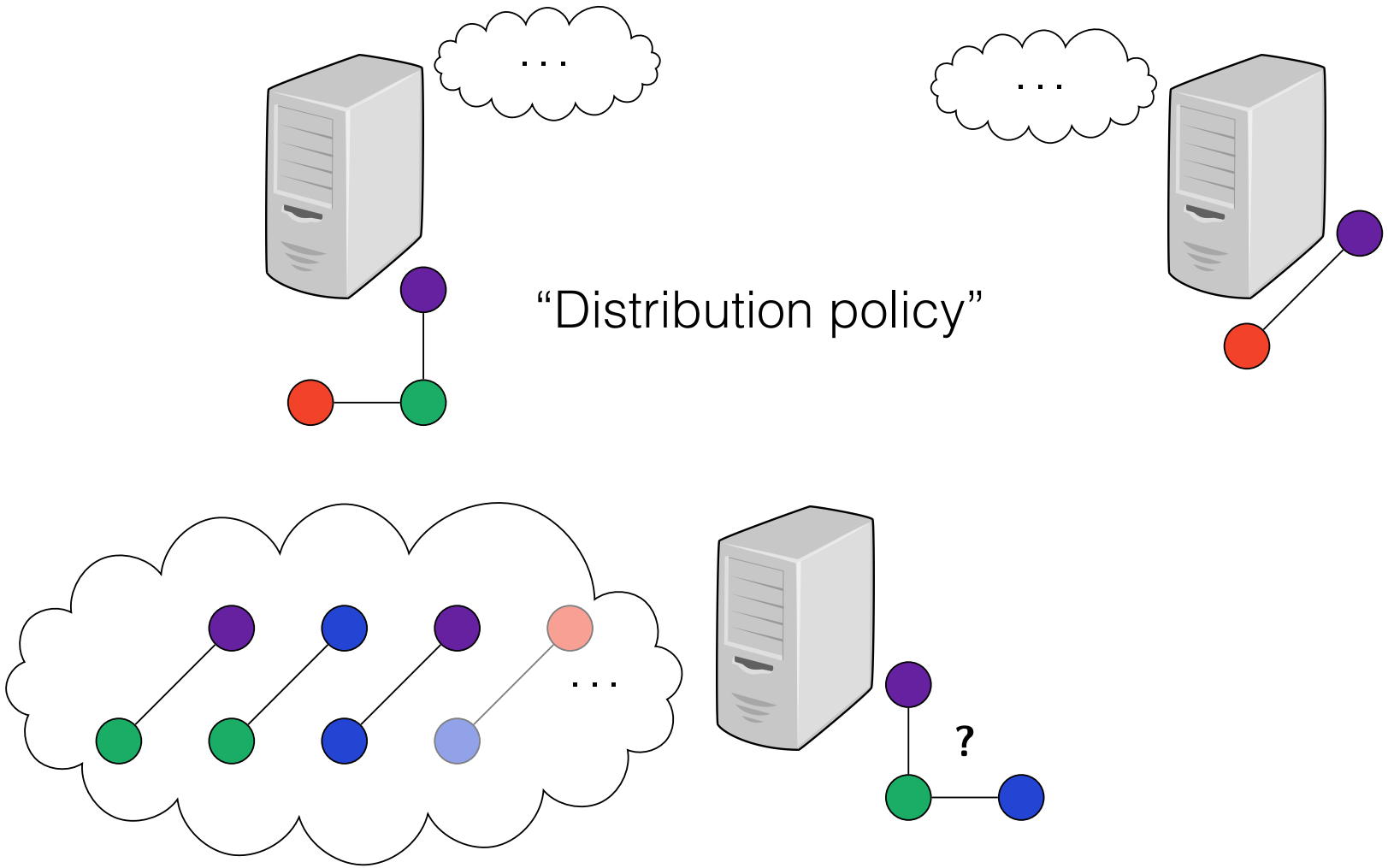


Policy-aware Transducers

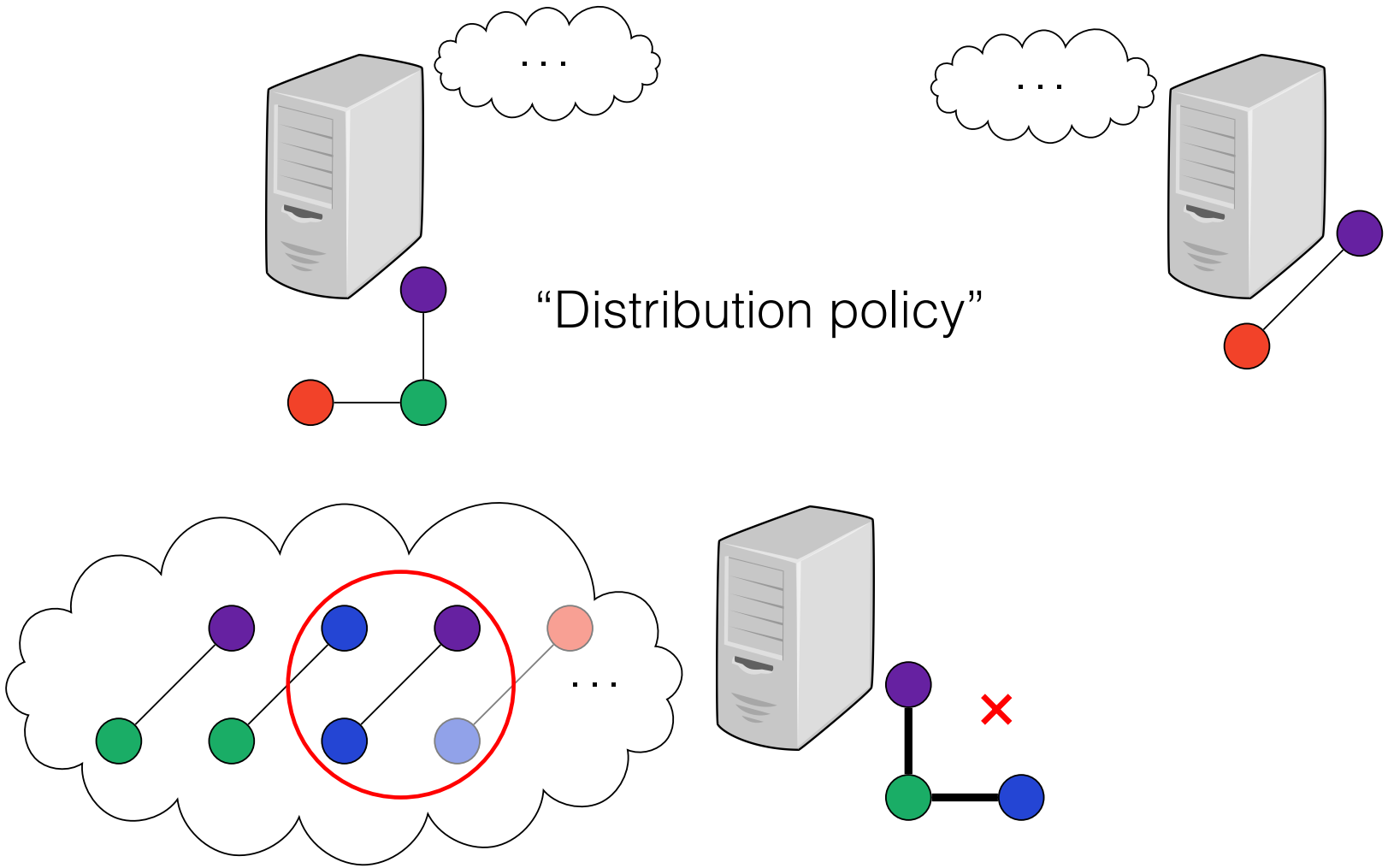
Deduction rules

- in local database $\Rightarrow$ in global database
- not in local database + in scope $\Rightarrow$ not in global database
- not in local database + not in scope $\Rightarrow$ unknown

Policy-aware Transducers



Policy-aware Transducers



Policy-aware Transducers

[Zinn, Green, Ludäscher, 2012]

Definition

A distribution policy $\mathbf{P}$ for σ and $\mathcal{N}$ is a total function from $facts(\sigma)$ to the power set of $\mathcal{N}$.

Definition

A policy-aware transducer is a transducer with access to $\mathbf{P}$ restricted to its active domain

Definition

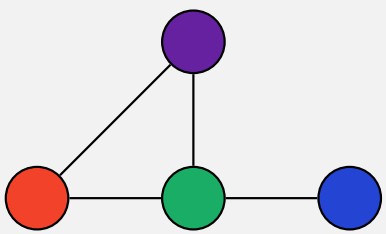
$\mathcal{F}_1 =$ set of queries which are distributedly computed by policy-aware coordination-free transducers

Domain-distinct-monotonicity

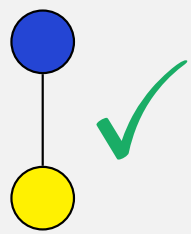
Definition

A fact **f** is domain distinct from instance $\mathbf{I}$ when $adom(\mathbf{f}) \not\subseteq adom(\mathbf{I})$.

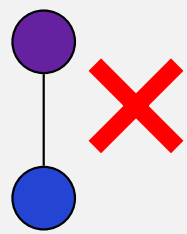
Example



I



f



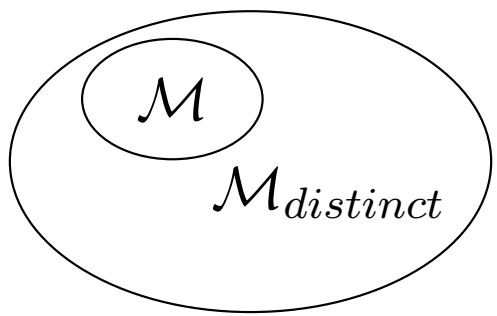
f'

Domain-distinct-monotonicity

Definition

A query $\mathcal{Q}$ is domain-distinct-monotone if $\mathcal{Q}(\mathbf{I}) \subseteq \mathcal{Q}(\mathbf{I} \cup \mathbf{J})$ for all $\mathbf{I}$ and $\mathbf{J}$, with $\mathbf{J}$ having only domain-distinct facts

Notation: $\mathcal{M}_{distinct}$ = domain-distinct-monotone queries



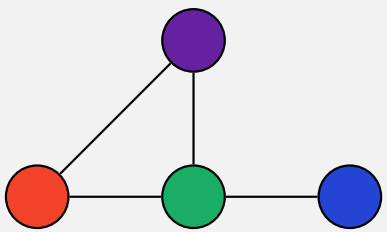
Remark

$\mathcal{M}_{distinct}$: class of queries preserved under extensions

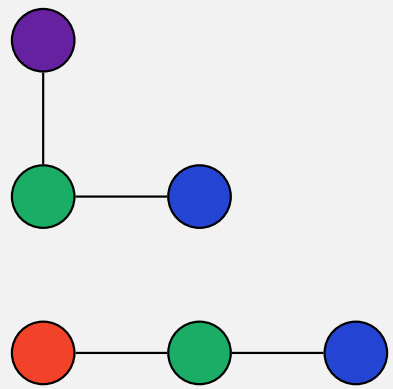
Domain-distinct-monotonicity

Example

Select open triangles in graph $\in \mathcal{M}_{distinct}$.



I

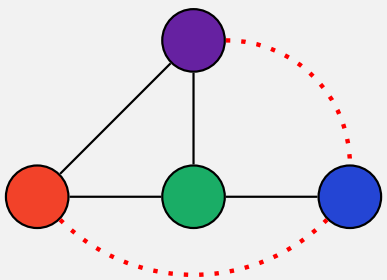


$\mathcal{Q}(I)$

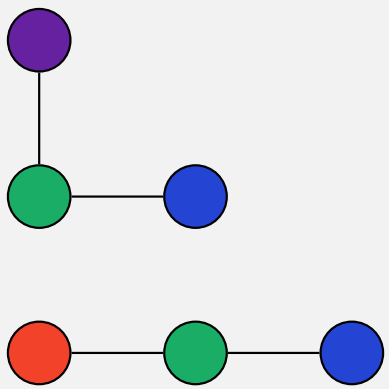
Domain-distinct-monotonicity

Example

Select open triangles in graph $\in \mathcal{M}_{distinct}$.



I

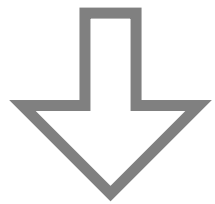


$\mathcal{Q}(I)$

Not domain-distinct from $\mathbf{I}$

Revised CALM-conjecture

A query has a coordination-free and eventually consistent execution strategy under distribution policies
iff
the query is domain-distinct-monotone



Theorem

$\mathcal{F}_1 = \mathcal{M}_{distinct}$

Definition

$\mathcal{F}_1 =$ set of queries which are distributedly computed by policy-aware coordination-free transducers

Proof idea of $\mathcal{M}_{\text{distinct}} \subseteq \mathcal{F}_1$

For domain-distinct-monotone queries:

- broadcast all present and deduced absent facts
- Evaluate query on complete sets

Proof idea of $\mathcal{M}_{\text{distinct}} \subseteq \mathcal{F}_1$

For domain-distinct-monotone queries:

- broadcast all present and deduced absent facts
- Evaluate query on complete sets

Example: $Q_<$: Select open triangles in a graph

Computing node behaviour:

Proof idea of $\mathcal{M}_{\text{distinct}} \subseteq \mathcal{F}_1$

For domain-distinct-monotone queries:

- broadcast all present and deduced absent facts
- Evaluate query on complete sets

Example: $Q_<$: Select open triangles in a graph

Computing node behaviour:

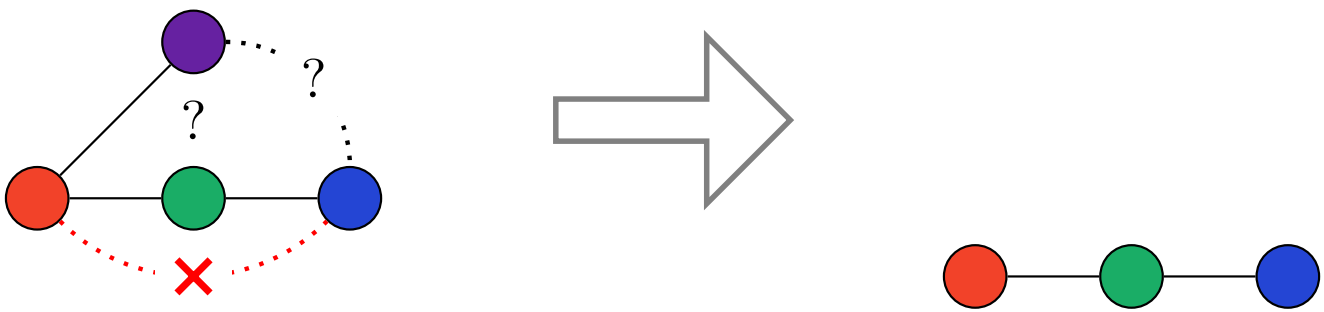
Proof idea of $\mathcal{M}_{\text{distinct}} \subseteq \mathcal{F}_1$

For domain-distinct-monotone queries:

- broadcast all present and deduced absent facts
- Evaluate query on complete sets

Example: $Q_<$: Select open triangles in a graph

Computing node behaviour:



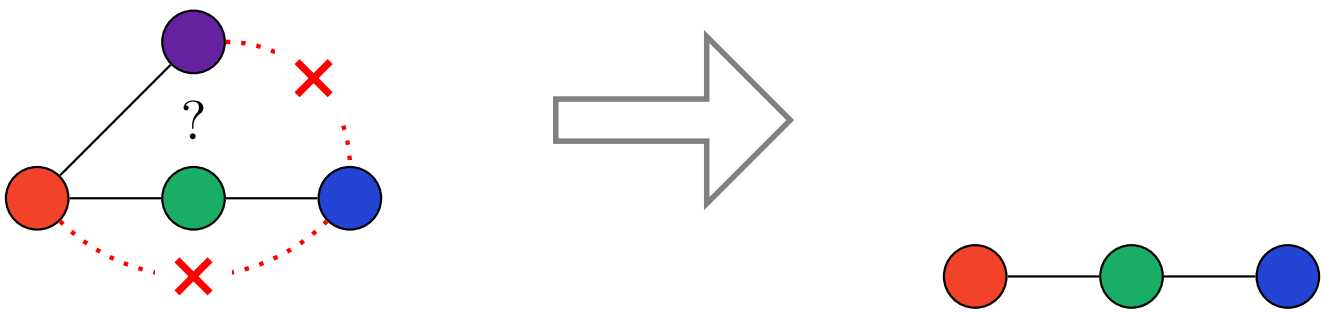
Proof idea of $\mathcal{M}_{\text{distinct}} \subseteq \mathcal{F}_1$

For domain-distinct-monotone queries:

- broadcast all present and deduced absent facts
- Evaluate query on complete sets

Example: $Q_<$: Select open triangles in a graph

Computing node behaviour:



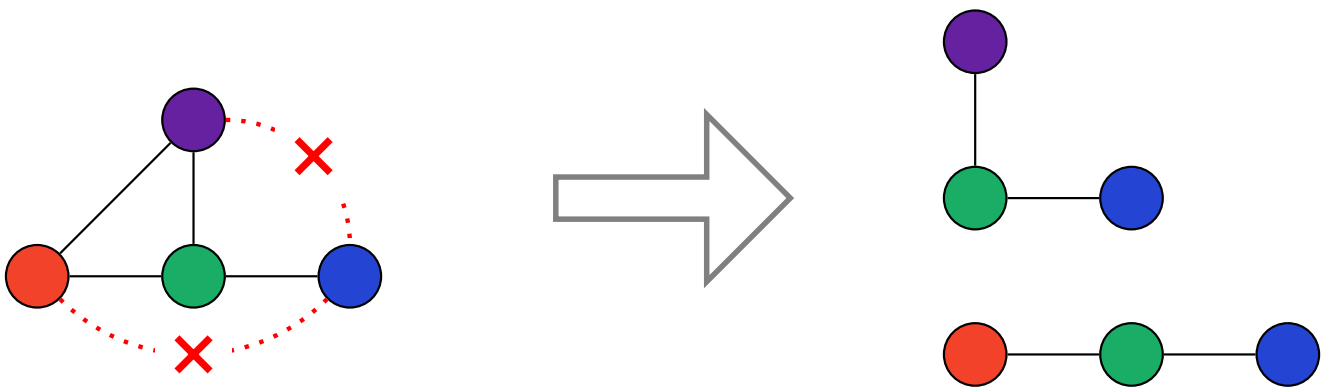
Proof idea of $\mathcal{M}_{\text{distinct}} \subseteq \mathcal{F}_1$

For domain-distinct-monotone queries:

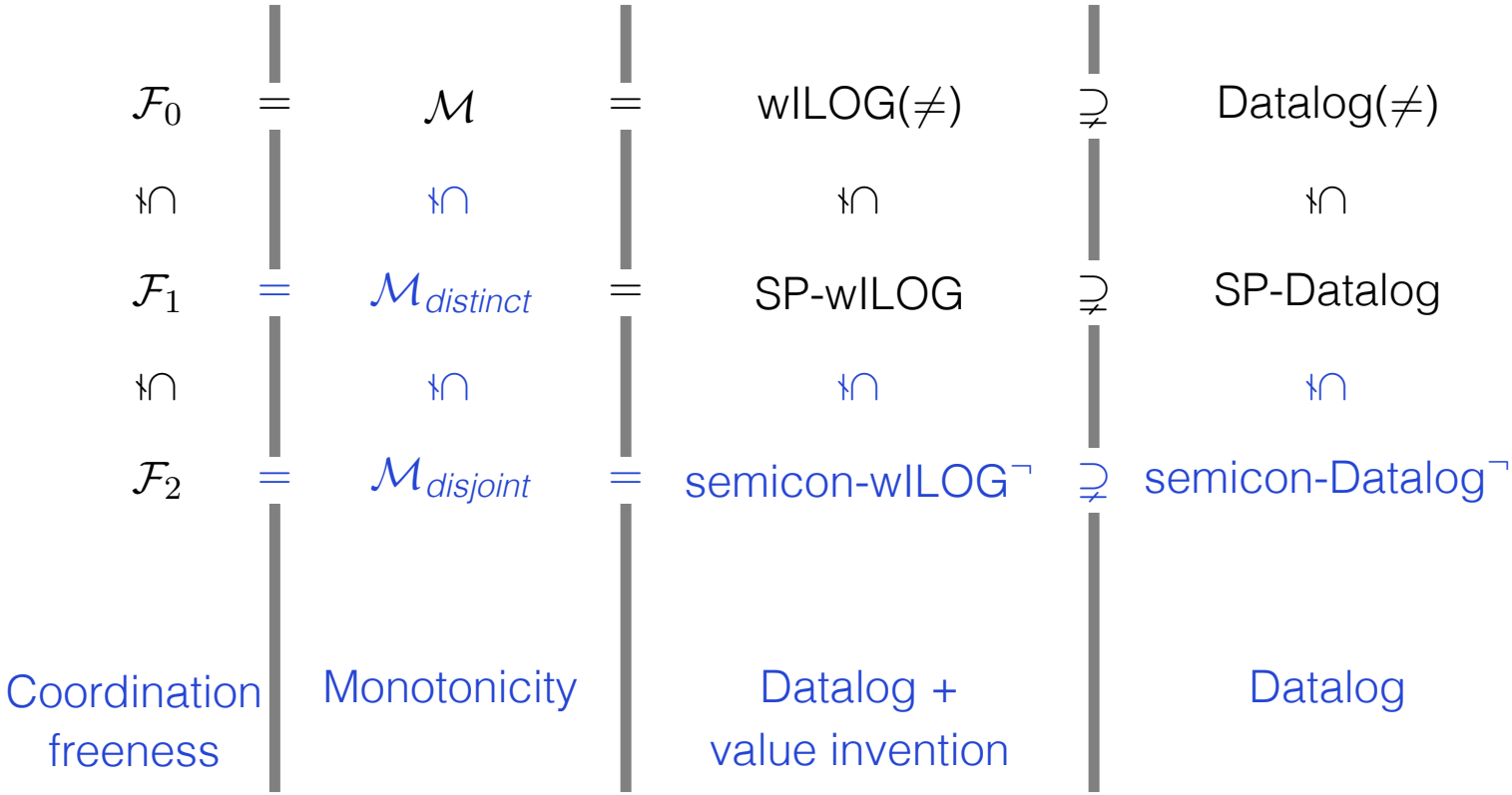
- broadcast all present and deduced absent facts
- Evaluate query on complete sets

Example: $Q_<$: Select open triangles in a graph

Computing node behaviour:



Complete Picture



[Ameloot, Ketsman, Neven, Zinn PODS 2014 **best paper**; TODS 2016]

Outline

CALM Formalization

CALM Revision 1

Conclusion

} Coordination-free evaluation

Parallel-Correctness

Transferability

Conclusion

} Multi-Query optimization

Goal: To clarify the relation between **monotonicity** and **coordination** in asynchronous systems and to reveal the more complete picture

---

- A four-level quantification of coordination exists in terms of the amount of information needed for the query to become coordination-free
- The CALM conjecture reveals a very robust relation between coordination and non-monotonic behaviour of queries

Open Questions & Future Work

- How to compute queries without coordination and with less communication?
- How to compute queries with “some” coordination?

Open Questions & Future Work

- How to compute queries without coordination and with less communication?
- How to compute queries with “some” coordination?

Related Work

- Oblivious broadcasting algorithms = broadcast fragment of local database [Ketsman, Neven ICDT 2015; ToCS 2016]
- A worst-case optimal load algorithm for join evaluation [Ketsman, Suciu, PODS 2017]

Outline

CALM Formalization

CALM Revision 1

Conclusion

} Coordination-free evaluation

Parallel-Correctness

Transferability

Conclusion

} Multi-Query optimization

Motivation

- Many systems rely on coordination for communication
  For example: MapReduce-like systems
- Avoiding coordination completely is not possible
- Minimize the number of communication steps
  Ideally: one round

Single-Round Evaluation Algorithm

(1-Round MPC model [Koutris & Suciu 2011])

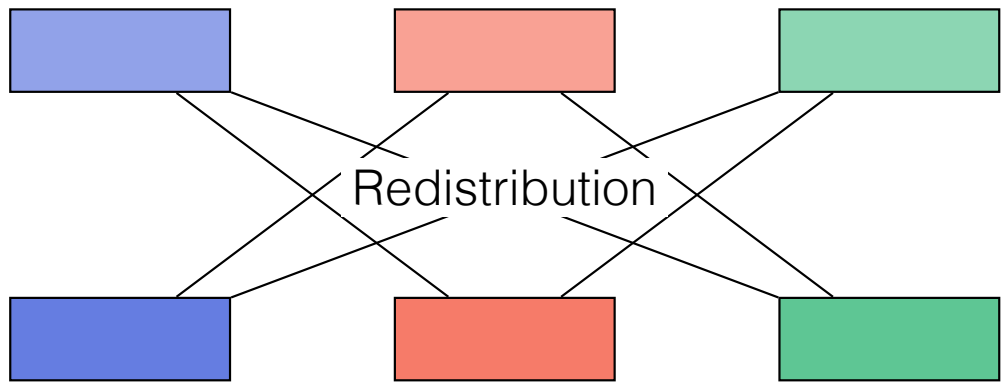
Input = query $\mathcal{Q}$

Single-Round Evaluation Algorithm

(1-Round MPC model [Koutris & Suciu 2011])

Input = query $\mathcal{Q}$

Step 1:



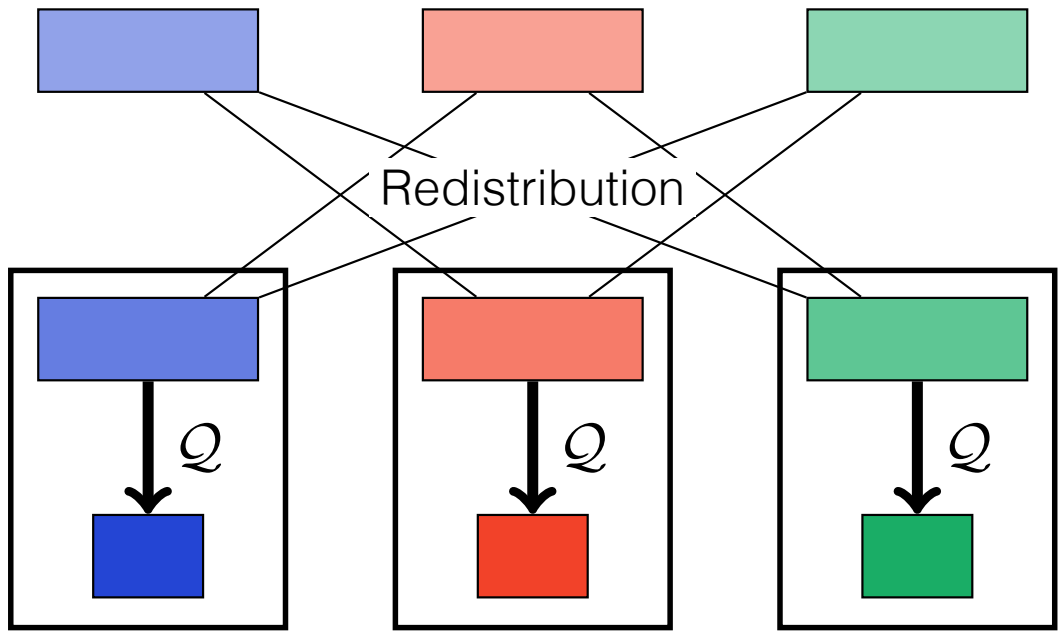
Single-Round Evaluation Algorithm

(1-Round MPC model [Koutris & Suciu 2011])

Input = query $\mathcal{Q}$

Step 1:

Step 2:



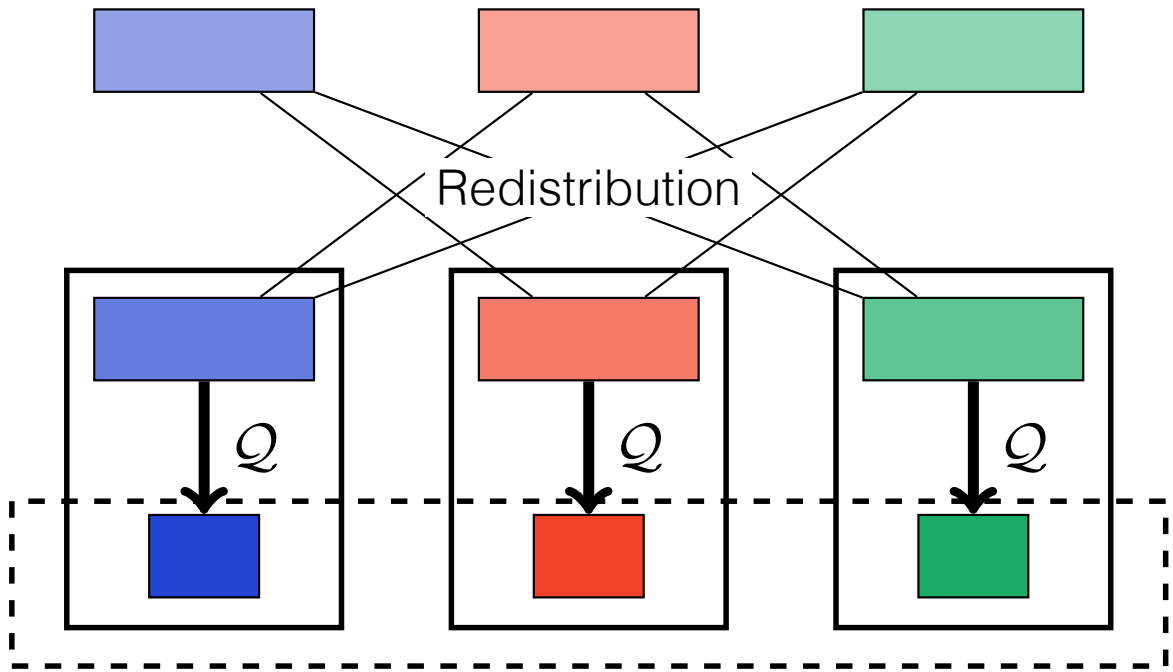
Single-Round Evaluation Algorithm

(1-Round MPC model [Koutris & Suciu 2011])

Input = query $\mathcal{Q}$

Step 1:

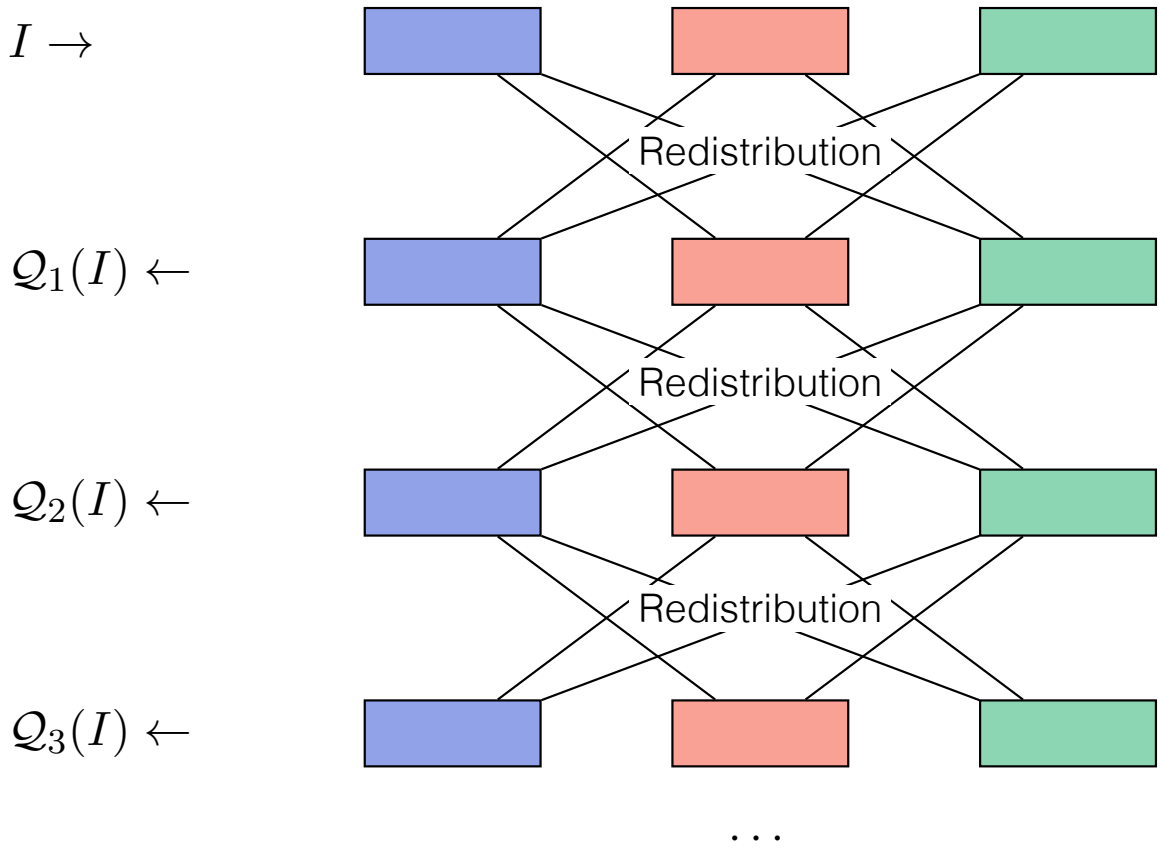
Step 2:



Output = union of output at each server

Multi-Query Evaluation & optimization

Workload: $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_n$ + fixed database



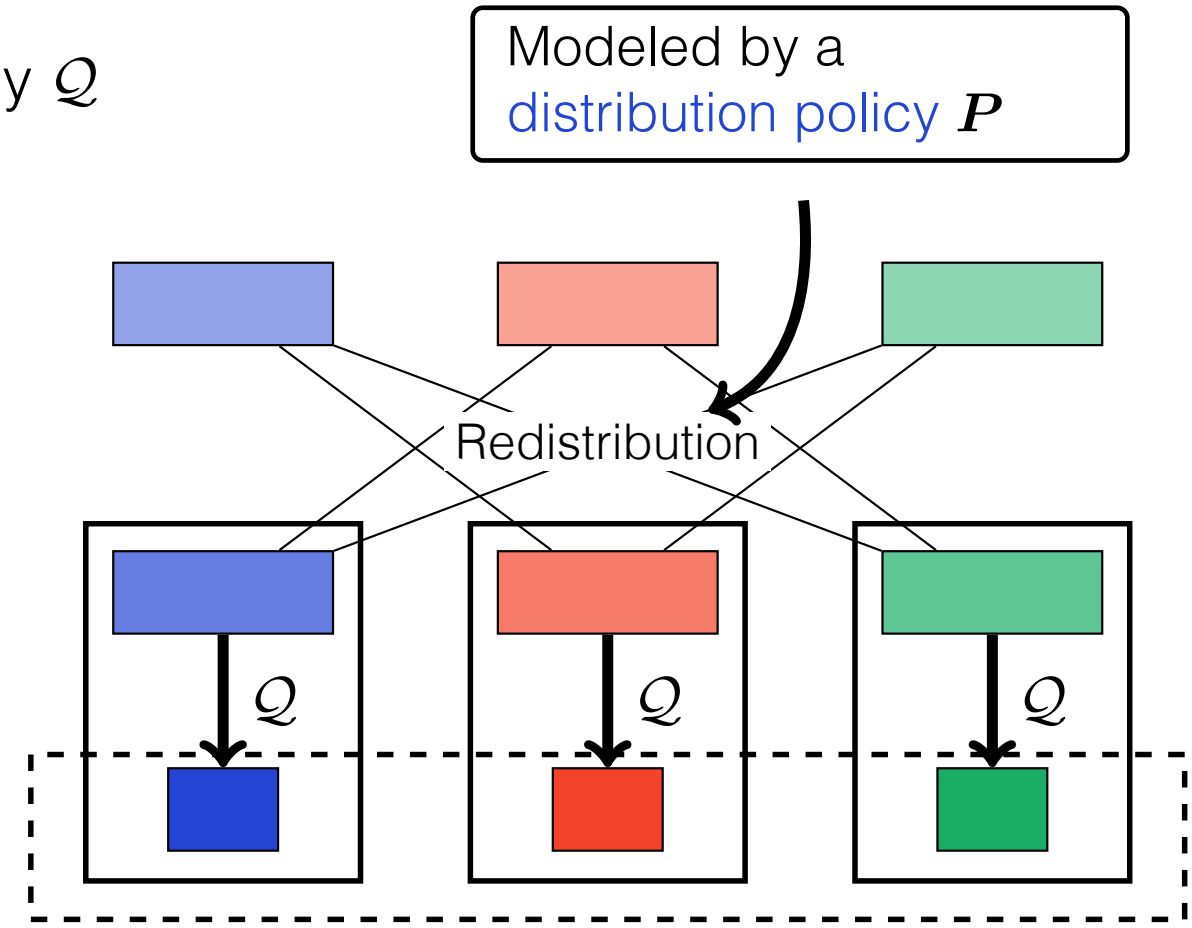
Goal: To formally reason about **single-round** query evaluation and **multi-query optimization** in systems where communication implies a synchronization barrier

Focus: conjunctive queries

Single-Round Evaluation Algorithm

(1-Round MPC model [Koutris & Suciu 2011])

Input = query Q



Output = union of output at each server

Can a reshuffle step be avoided?

Main question 1

Given target query and distribution policy:

Do we need to reshuffle?

Can a reshuffle step be avoided?

Main question 1

Given target query and distribution policy:

Do we need to reshuffle?

Parallel-Correctness

Can a reshuffle step be avoided?

Main question 1

Given target query and distribution policy:

Do we need to reshuffle?

Parallel-Correctness

Main question 2

Given target query and previously computed query:

Do we need to reshuffle?

Can a reshuffle step be avoided?

Main question 1

Given target query and distribution policy:

Do we need to reshuffle?

Parallel-Correctness

Main question 2

Given target query and previously computed query:

Do we need to reshuffle?

Transferability

Outline

CALM Formalization

CALM Revision 1

Conclusion

} Coordination-free evaluation

Parallel-Correctness

Transferability

Conclusion

} Multi-Query optimization

Parallel-Correctness

Given target query and distribution policy:

Do we need to reshuffle?

Parallel-Correctness

Given target query and distribution policy:

Do we need to reshuffle?

Semantical correctness of simple evaluation algorithm

Parallel-Correctness

Given target query and distribution policy:

Do we need to reshuffle?

Semantical correctness of simple evaluation algorithm

Definition

$\mathcal{Q}$ is parallel-correct w.r.t. $\boldsymbol{P}$, iff for every database I

$$\mathcal{Q}(I) = \bigcup_{\kappa \in \mathcal{N}} \mathcal{Q}(dist_{\boldsymbol{P},I}(\kappa))$$

Parallel-Correctness

Given target query and distribution policy:

Do we need to reshuffle?

Semantical correctness of simple evaluation algorithm

Definition

$\mathcal{Q}$ is parallel-correct w.r.t. $\boldsymbol{P}$, iff for every database I

$$\mathcal{Q}(I) = \bigcup_{\kappa \in \mathcal{N}} \mathcal{Q}(dist_{\boldsymbol{P},I}(\kappa))$$

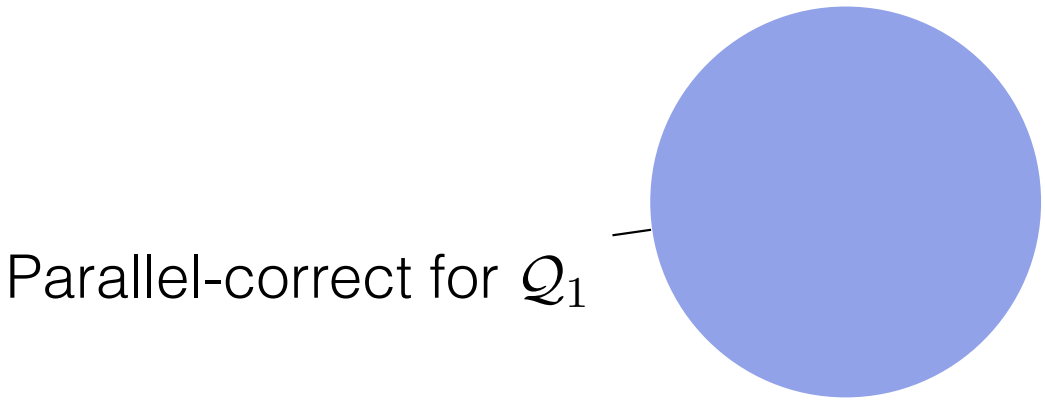
$\supseteq$ by monotonicity

Parallel-Correctness Complexity

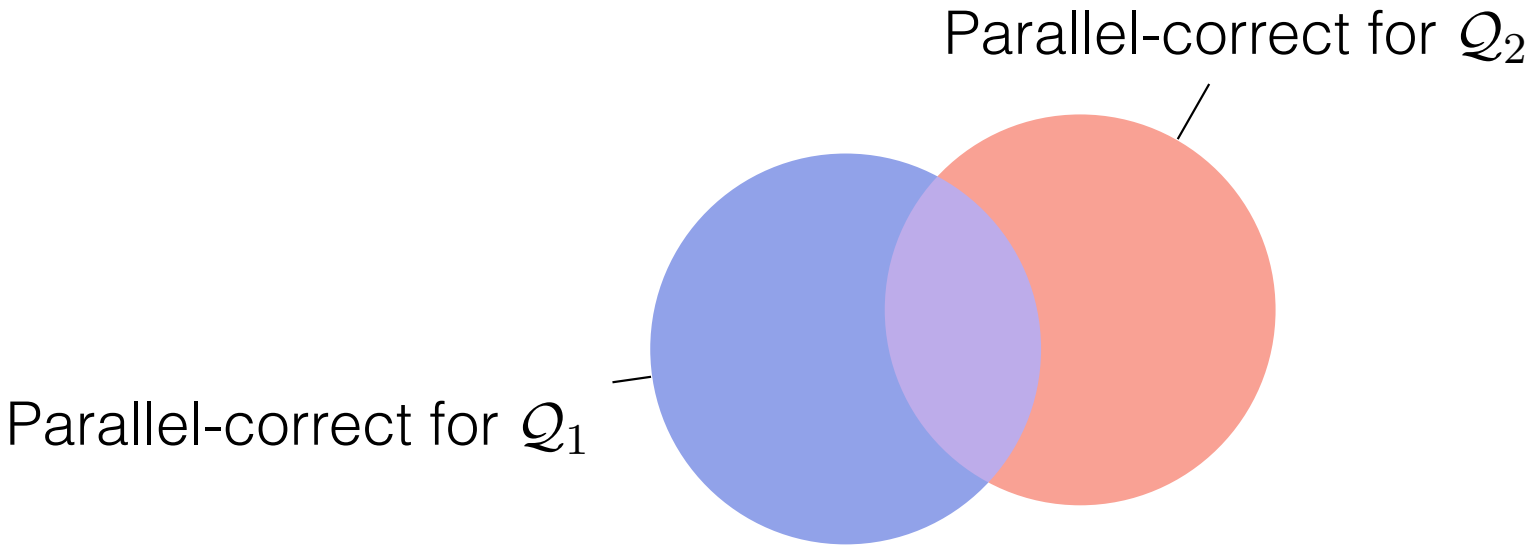
CQ	Π_p^2-c
UCQ	Π_p^2-c
UCQ$^{\neq}$	Π_p^2-c
FO	undecidable

(needs policy representation)

Other use of Parallel-Correctness



Other use of Parallel-Correctness



Possible Problems

- Reasoning about distribution policies is complex
- Not every distribution policy is equally efficient
- Choice of policy may be hidden behind abstraction layer
- Reasoning about query order before policies are known

Outline

CALM Formalization

CALM Revision 1

Conclusion

} Coordination-free evaluation

Parallel-Correctness

Transferability

Conclusion

} Multi-Query optimization

Transferability

Given target query and previously computed query:

Do we need to reshuffle?

Transferability

Given target query and previously computed query:

Do we need to reshuffle?

Definition

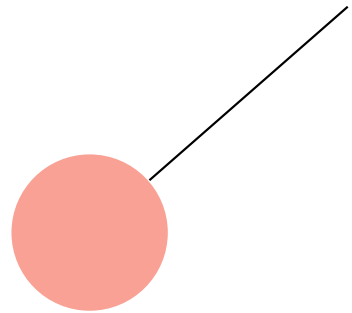
$\mathcal{Q} \rightarrow_T \mathcal{Q}'$ iff $\mathcal{Q}'$ is parallel-correct on every $\boldsymbol{P}$ where $\mathcal{Q}$ is parallel-correct on

Transferability

Given target query and previously computed query:

Do we need to reshuffle?

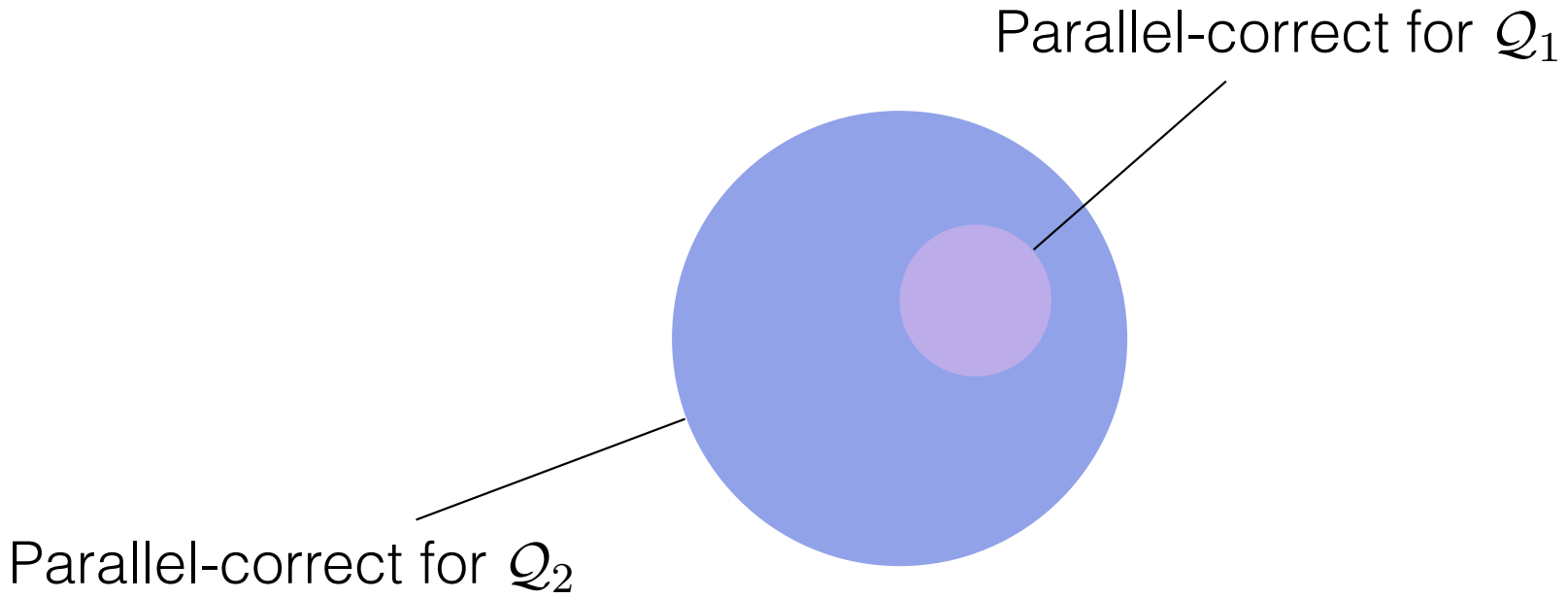
Parallel-correct for $\mathcal{Q}_1$



Transferability

Given target query and previously computed query:

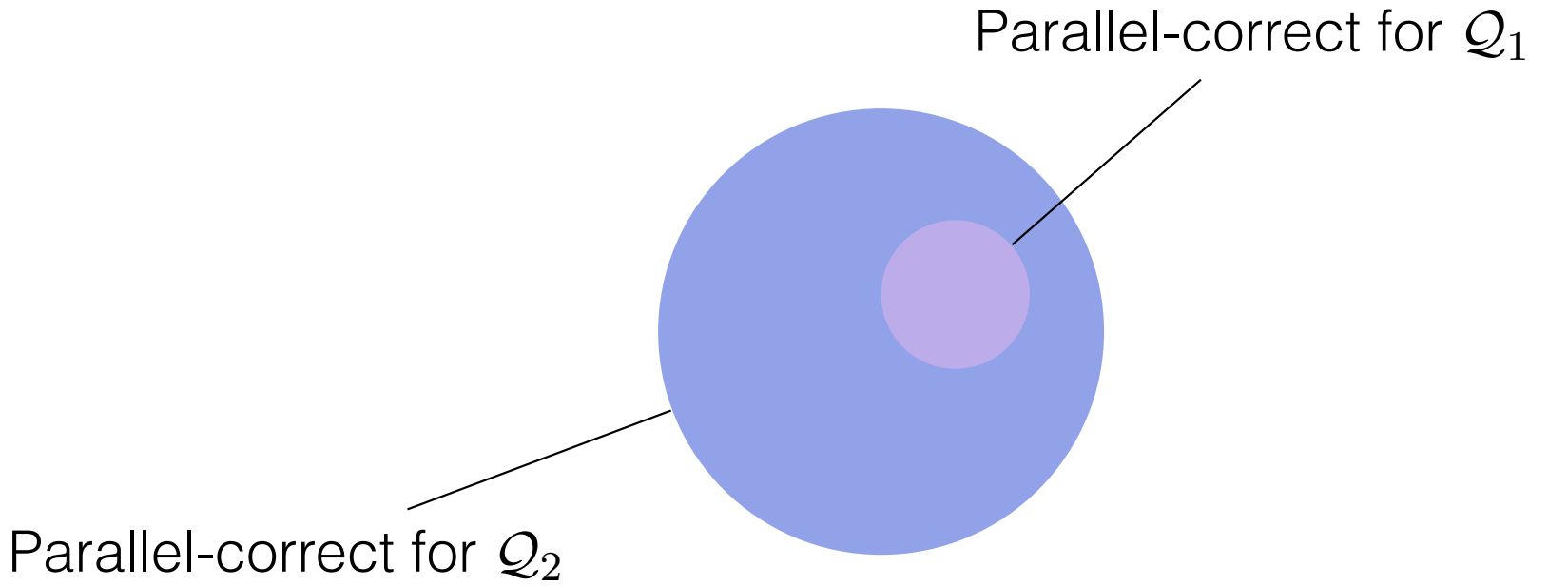
Do we need to reshuffle?



Transferability

Given target query and previously computed query:

Do we need to reshuffle?



Very strong property

like query containment, but for parallel and distributed setting

Complexity

	Parallel-correctness*	Transferability
CQ	Π_p^2-c	Π_p^3-c
UCQ	Π_p^2-c	Π_p^3-c
UCQ$^{\neq}$	Π_p^2-c	Π_p^3-c
FO	undecidable	undecidable
sm-CQ	X	NP-c

(*needs policy representation)

[Ameloot,Geck,Ketsman,Neven,Schwentick PODS 2015 **best paper**;
Sigmod record 2016; CACM 2017; JACM (accepted)]

Outline

CALM Formalization

CALM Revision 1

Conclusion

} Coordination-free evaluation

Parallel-Correctness

Transferability

Conclusion

} Multi-Query optimization

Goal: To formally reason about **single-round** query evaluation and **multi-query optimization** in systems where communication implies a synchronization barrier

---

- A formal framework for reasoning about the correctness of single-round query evaluation and query optimization via distribution policies.
- **Parallel-correctness**: semantical correctness
- **Transferability**: like “containment” but for parallel and distributed query evaluation

Open Questions & Future Work

- How do parallel-correctness and transferability relate to query evaluation in practice?
- How much data needs to be reshuffled?

Open Questions & Future Work

- How do parallel-correctness and transferability relate to query evaluation in practice?
- How much data needs to be reshuffled?

Related Work

- Parallel-correctness for CQs with negation

  [Geck, Ketsman, Neven, Schwentick ICDT 2016]

- Extension of Parallel-correctness to reason about multi-round evaluation with with Datalog

  [Ketsman, Koutris, Albarghouti, submitted]

- Bag-semantics

  ongoing

Thank you!