Parallel-Correctness and Parallel-Boundedness for Datalog Programs

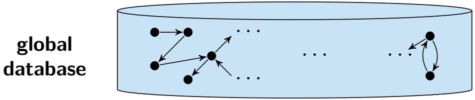
Frank Neven[1] Thomas Schwentick[2] Christopher Spinrath[2] Brecht Vandevoort[1]

[1]Hasselt University and transnational University of Limburg

[2]TU Dortmund University

22nd ICDT March 28, 2019 Lisbon

Distributed Evaluation



Query

- transitive closure T
- Datalog program

$$T(x,y) \leftarrow E(x,y).$$
$$T(x,z) \leftarrow T(x,y),\ E(y,z).$$

- recursive evaluation
  (fixed point computation)

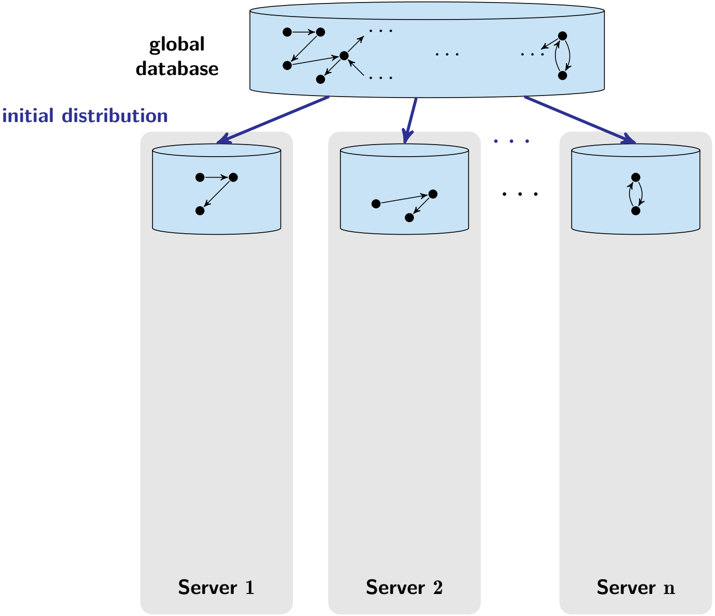
Distributed Evaluation

Query

- transitive closure T
- Datalog program

$$T(x,y) \leftarrow E(x,y).$$
$$T(x,z) \leftarrow T(x,y),\ E(y,z).$$

- recursive evaluation (fixed point computation)



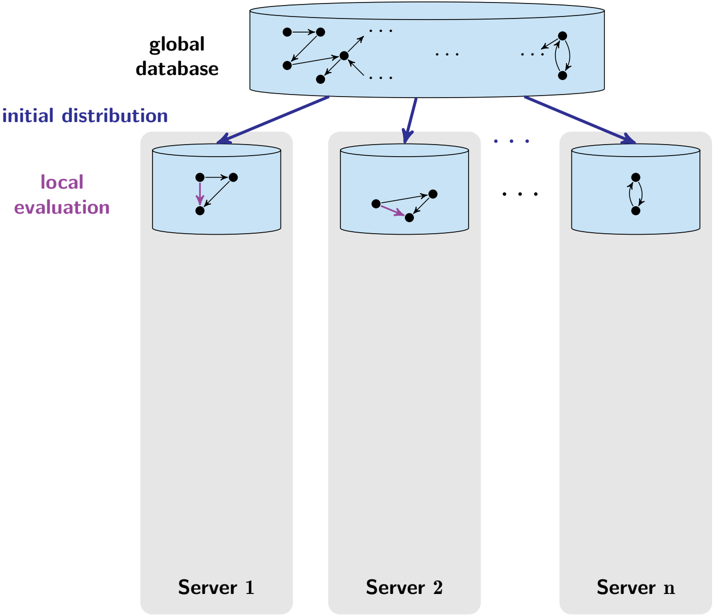
Distributed Evaluation

Query

- transitive closure T
- Datalog program

$$T(x,y) \leftarrow E(x,y).$$
$$T(x,z) \leftarrow T(x,y),\ E(y,z).$$

- recursive evaluation (fixed point computation)



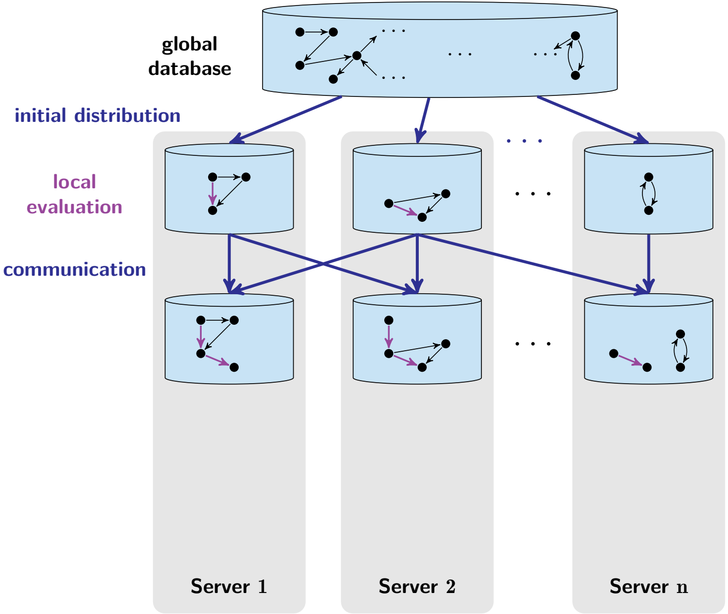
Distributed Evaluation

Query

- transitive closure T
- Datalog program

$$T(x,y) \leftarrow E(x,y).$$
$$T(x,z) \leftarrow T(x,y),\ E(y,z).$$

- recursive evaluation (fixed point computation)



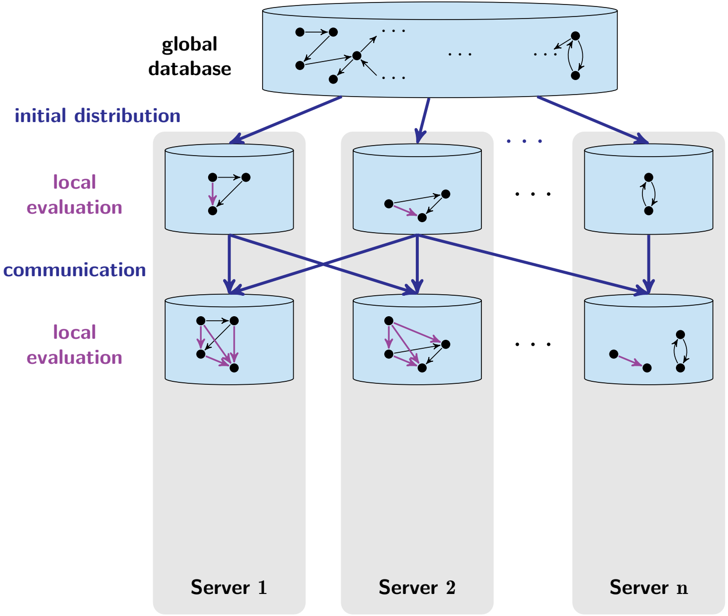
Distributed Evaluation

Query

- transitive closure T
- Datalog program

$$T(x,y) \leftarrow E(x,y).$$
$$T(x,z) \leftarrow T(x,y),\ E(y,z).$$

- recursive evaluation (fixed point computation)



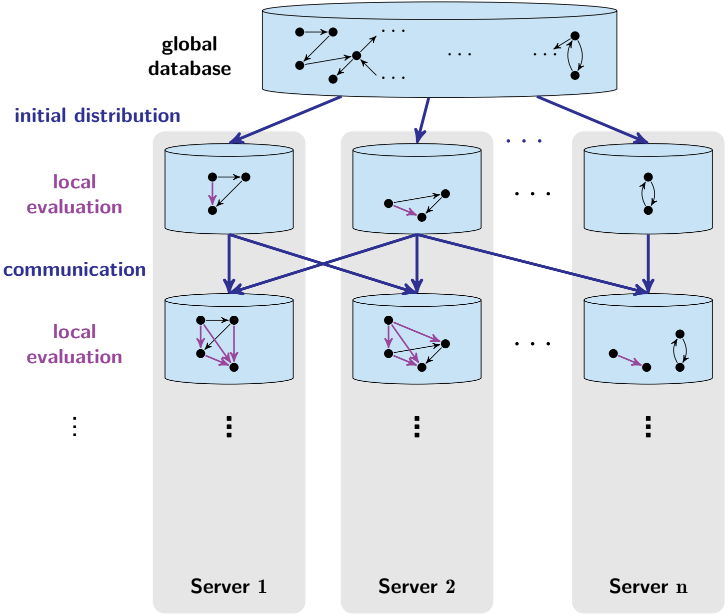
Distributed Evaluation

Query

- transitive closure T
- Datalog program

$$T(x,y) \leftarrow E(x,y).$$
$$T(x,z) \leftarrow T(x,y),\ E(y,z).$$

- recursive evaluation (fixed point computation)



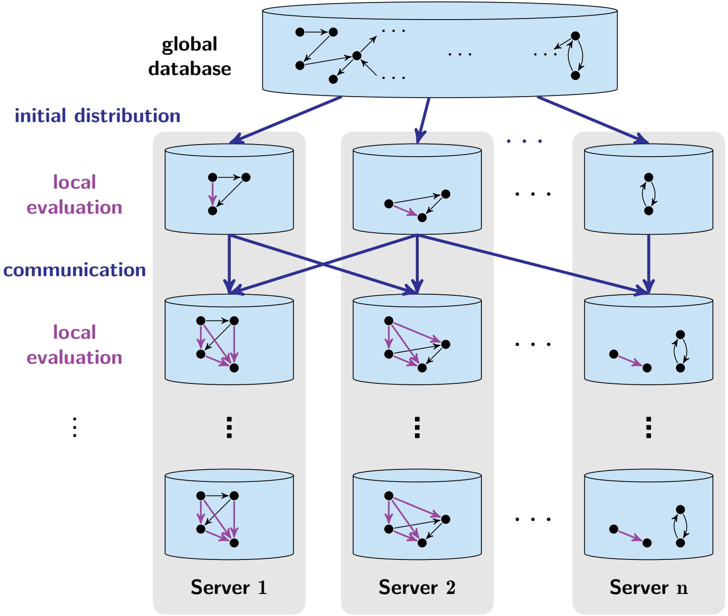
Distributed Evaluation

Query

- transitive closure T
- Datalog program

$$T(x, y) \leftarrow E(x, y).$$
$$T(x, z) \leftarrow T(x, y),\ E(y, z).$$

- recursive evaluation (fixed point computation)



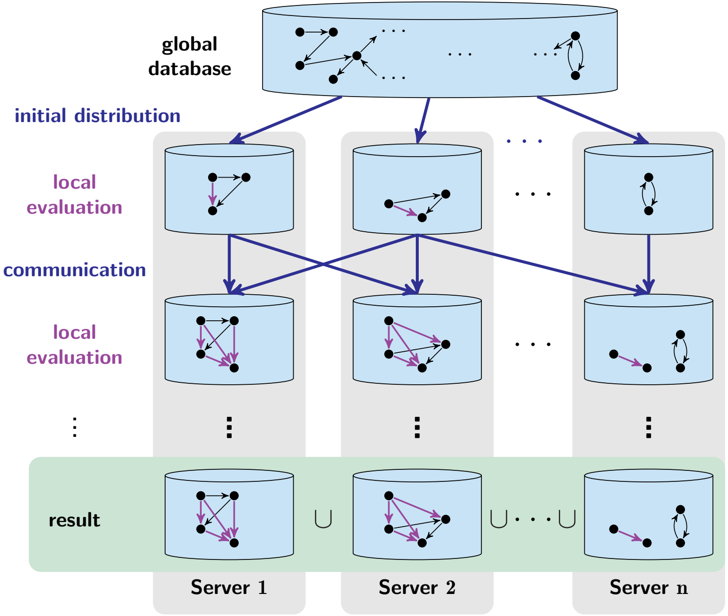
Distributed Evaluation

Query

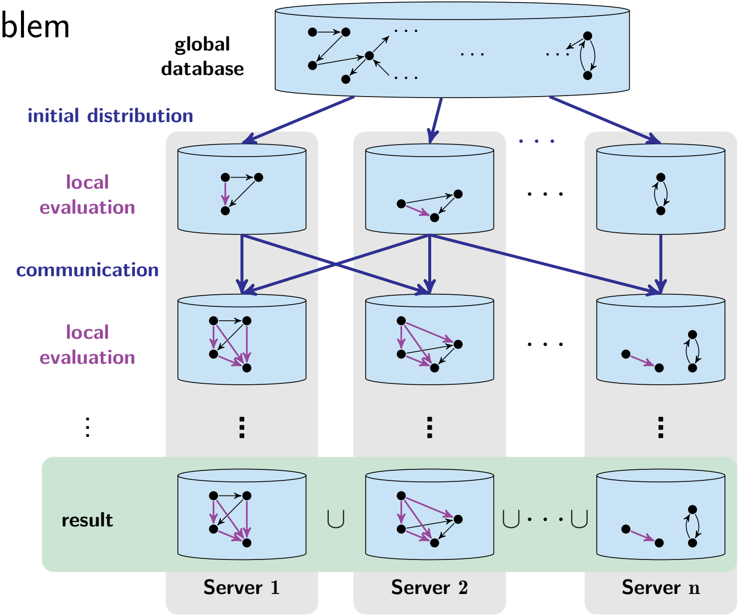
- transitive closure T
- Datalog program

$$T(x,y) \leftarrow E(x,y).$$
$$T(x,z) \leftarrow T(x,y),\ E(y,z).$$

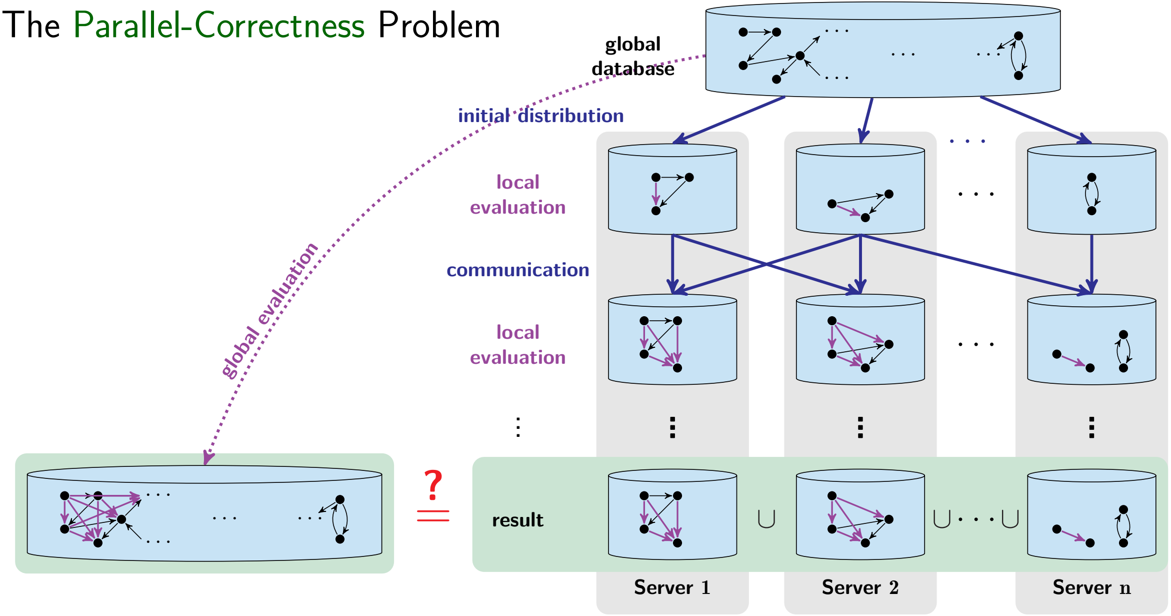
- recursive evaluation (fixed point computation)



The Parallel-Correctness Problem



The Parallel-Correctness Problem



The Parallel-Correctness Problem

Parallel-Correctness Problem

Input:

- Datalog program
- distribution policy
- communication policy

Question:

Do distributed and global evaluation yield the same result for all databases?

The Parallel-Correctness Problem

Parallel-Correctness Problem

Input:

- Datalog program
- distribution policy
- communication policy

Question:

Do distributed and global evaluation yield the same result for all databases?

Theorem (Ketsman, Albarghouthi, and Koutris 2018)

Parallel-correctness for general Datalog programs is undecidable.

The Parallel-Correctness Problem

Parallel-Correctness Problem

Input:

- Datalog program
- distribution policy
- communication policy

Question:

Do distributed and global evaluation yield the same result for all databases?

Theorem (Ketsman, Albarghouthi, and Koutris 2018)

Parallel-correctness for general Datalog programs is undecidable.

- Even for “simple” policies:
  - only two servers
  - all but one relations are distributed to both servers
  - no communication

The Parallel-Correctness Problem

Parallel-Correctness Problem

Input:

- Datalog program
- distribution policy
- communication policy

Question:

Do distributed and global evaluation yield the same result for all databases?

Theorem (Ketsman, Albarghouthi, and Koutris 2018)

Parallel-correctness for general Datalog programs is undecidable.

- Even for "simple" policies:
  - only two servers
  - all but one relations are distributed to both servers
  - no communication
- Is there a fragment of Datalog for which parallel-correctness is decidable?

Basics

Relational databases

$\underbrace{E(a,b,c)}_{\text{fact}}, E(a,d,g), F(a,d), \dots$

with extensional relation symbols $E, F, \dots$

Basics

Relational databases

$\underbrace{E(a,b,c)}_{\text{fact}}, E(a,d,g), F(a,d), \ldots$

with extensional relation symbols $E, F, \ldots$

Datalog programs consist of rules

$$\underbrace{T(x,y)}_{\text{head}} \leftarrow \underbrace{E(x,y,z), \overbrace{R(x,v)}^{\text{atom}}}_{\text{body}}.$$

head atoms are intensional (i.e. not extensional)

Parallel-Correctness and Containment

Undecidability of parallel-correctness results from the containment problem

… and containment is undecidable for general Datalog

Parallel-Correctness and Containment

Undecidability of parallel-correctness results from the containment problem

… and containment is undecidable for general Datalog

general Datalog

$\subsetneq$

monadic Datalog	only unary head atoms	Example: $R(x) \leftarrow S(x), E(y, z, u).$

Containment is decidable for monadic Datalog

Parallel-Correctness and Containment

Undecidability of parallel-correctness results from the containment problem

… and containment is undecidable for general Datalog

general Datalog

$\subsetneq$

frontier-guarded Datalog

Each rule has a **guard** atom

- contains all head variables
- extensional

Example:

$T(x, y) \leftarrow \mathbf{E}(\mathbf{x}, \mathbf{y}, \mathbf{z}), F(y, v).$

$\subsetneq$

monadic Datalog

only unary head atoms

Example:

$R(x) \leftarrow S(x), E(y, z, u).$

Containment is decidable for monadic Datalog and frontier-guarded Datalog

Parallel-Correctness and Containment

Undecidability of parallel-correctness results from the containment problem

... and containment is undecidable for general Datalog

general Datalog

frontier-guarded Datalog

Each rule has a **guard** atom
- contains all head variables
- extensional

Example:
$T(x, y) \leftarrow \mathbf{E}(\mathbf{x}, \mathbf{y}, \mathbf{z}), F(y, v).$

rewrite

monadic Datalog

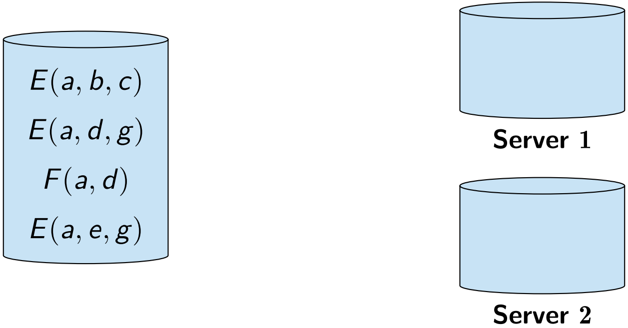
only unary head atoms

Example:
$R(x) \leftarrow S(x), E(y, z, u).$

Containment is decidable for monadic Datalog and frontier-guarded Datalog

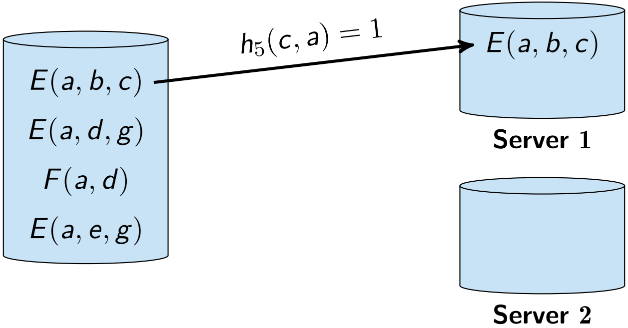
Distribution Policies

Idea: Use hash functions $h_1, \ldots, h_k$ fast, evenly distribution



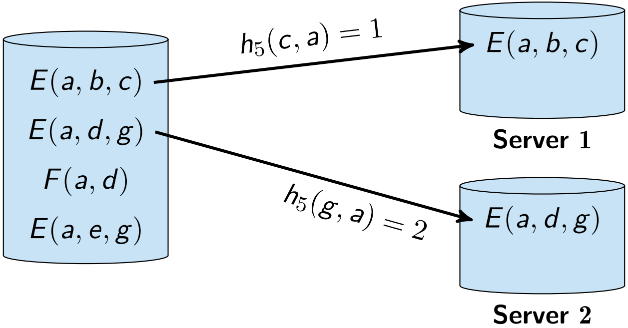
Distribution Policies

Idea: Use hash functions $h_1, \ldots, h_k$ fast, evenly distribution



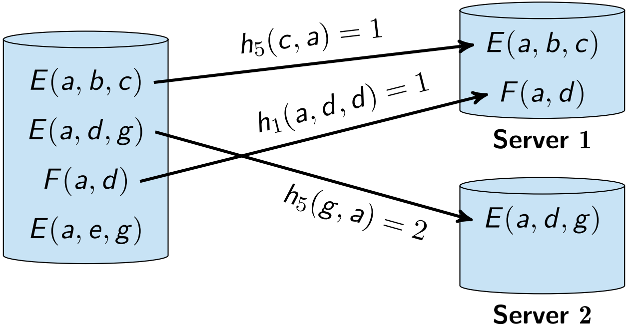
Distribution Policies

Idea: Use hash functions $h_1, \ldots, h_k$ fast, evenly distribution



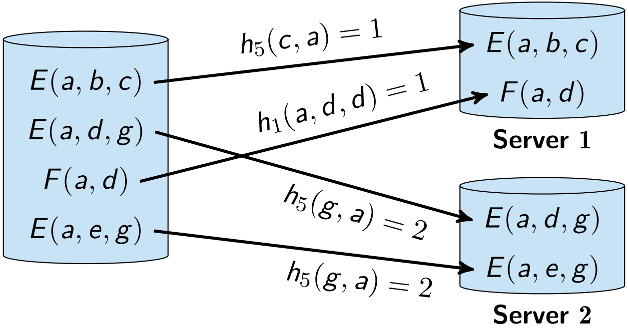
Distribution Policies

Idea: Use hash functions $h_1, \ldots, h_k$ fast, evenly distribution



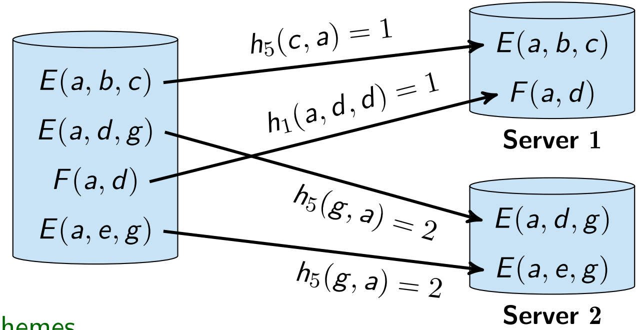
Distribution Policies

Idea: Use hash functions $h_1, \ldots, h_k$ fast, evenly distribution



Distribution Policies

Idea: Use hash functions $h_1, \dots, h_k$ fast, evenly distribution



Here: Hash policy schemes

- describes how hash functions are applied
- defines class of hash functions

Communication Policies

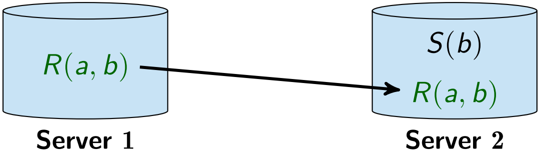
Data-Moving Distribution Constraints

$$\underbrace{R(x, y)@\lambda, S(y)@\kappa}_{\text{body}} \rightarrow \underbrace{R(x, y)@\kappa}_{\text{head}}$$

Communication Policies

Data-Moving Distribution Constraints

$$\underbrace{R(x,y)@\lambda, S(y)@\kappa}_{\text{body}} \rightarrow \underbrace{R(x,y)@\kappa}_{\text{head}}$$



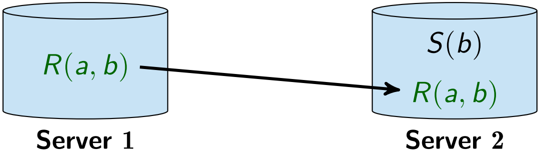
Communication Policies

Data-Moving Distribution Constraints

$$\underbrace{R(x,y)@\lambda, S(y)@\kappa}_{\text{body}} \rightarrow \underbrace{R(x,y)@\kappa}_{\text{head}}$$

Both $R(x,y)$ and κ occur in the body.

- No creation of facts
- No creation of servers



Parallel-Correctness

Theorem

Parallel-correctness for monadic and frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*

is undecidable.

A Decidable Variant

Theorem

Parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*
- *with the polynomial communication property*

is 2ExpTime-complete.

A Decidable Variant

Theorem

Parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*
- *with the polynomial communication property*

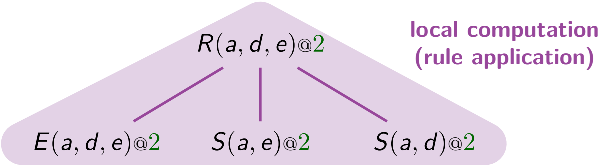
is 2ExpTime*-complete.*

Lower bound: reduction from the containment problem

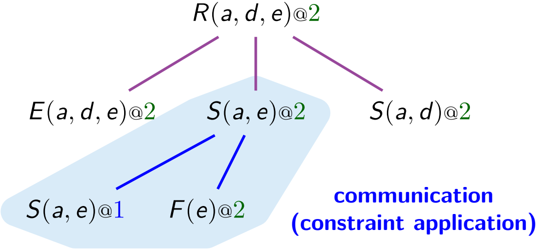
Distributed Proof Trees

$R(a, d, e)@2$

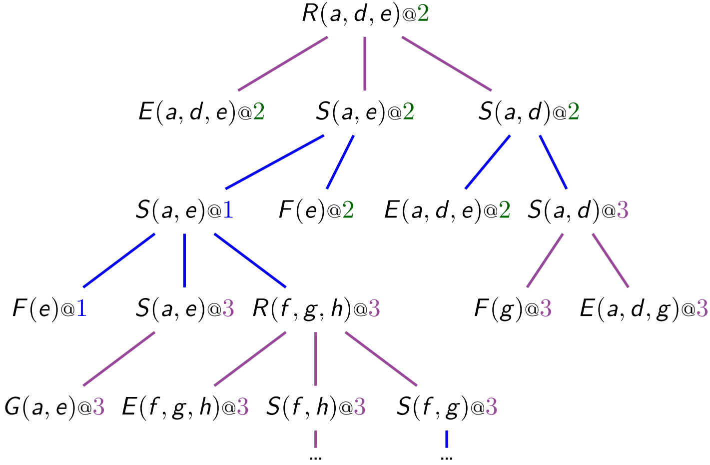
Distributed Proof Trees



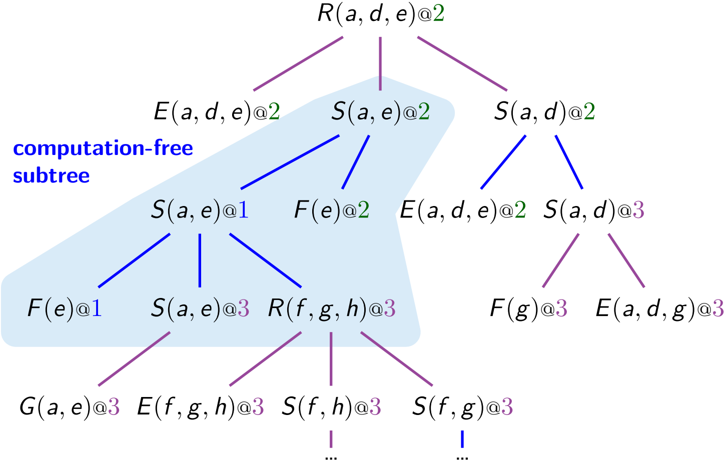
Distributed Proof Trees



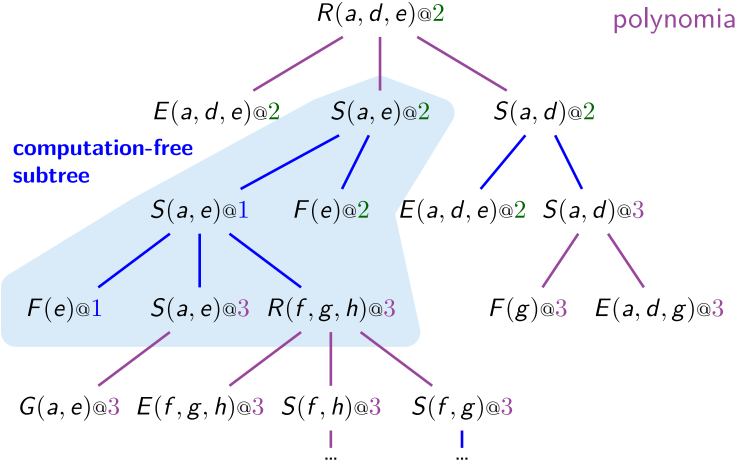
Distributed Proof Trees



Distributed Proof Trees



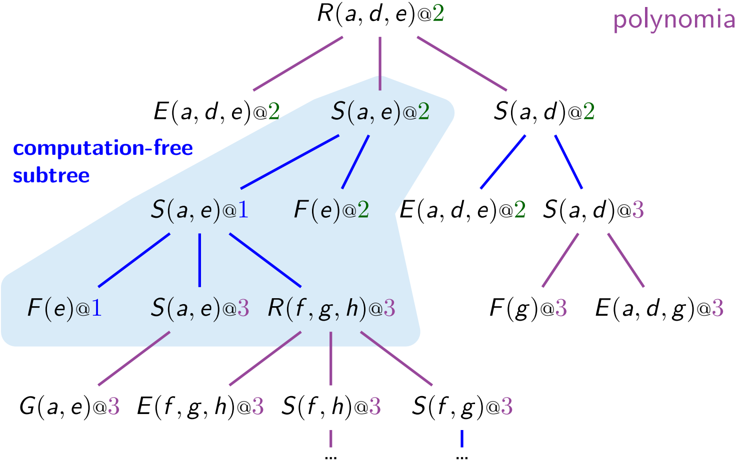
Distributed Proof Trees



Polynomial Communication Property

The size of computation-free subtrees is polynomially bounded.

Distributed Proof Trees



Polynomial Communication Property

The size of computation-free subtrees is polynomially bounded.

Two Settings

1. Syntactical restriction of distribution constraints
2. Changed semantics
   non-transitive setting

Proof Approach: Parallel-Correctness is in 2ExpTime

❶ Consider only worst-case hash functions

Proof Approach: Parallel-Correctness is in 2ExpTime

❶ Consider only worst-case hash functions

❷ Compile

- distribution policy,
- communication policy, and
- input Datalog program

into frontier-guarded Datalog program

Proof Approach: Parallel-Correctness is in 2ExpTime

1. Consider only worst-case hash functions

2. Compile
   - distribution policy,
   - communication policy, and
   - input Datalog program

   into frontier-guarded Datalog program

   that simulates worst-case distributed evaluation

Proof Approach: Parallel-Correctness is in 2ExpTime

1. Consider only worst-case hash functions
2. Compile
   - distribution policy,
   - communication policy, and
   - input Datalog program

   into frontier-guarded Datalog program

   that simulates worst-case distributed evaluation
3. Test containment

Proof Ingredient: Scattering Hash Functions

Two facts meet on the same server, if and only if,

- they were hashed by the same hash function, and
- according to the same values.

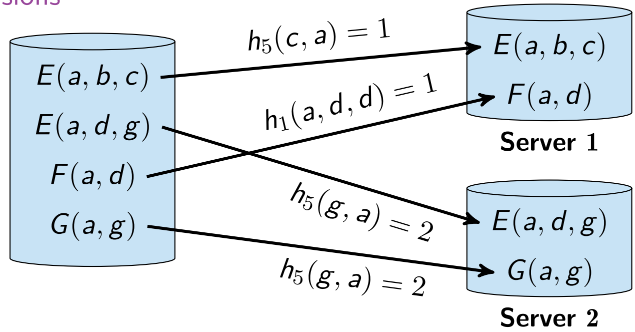
No "accidental" collisions

Proof Ingredient: Scattering Hash Functions

Two facts meet on the same server, if and only if,

- they were hashed by the same hash function, and
- according to the same values.

No “accidental” collisions

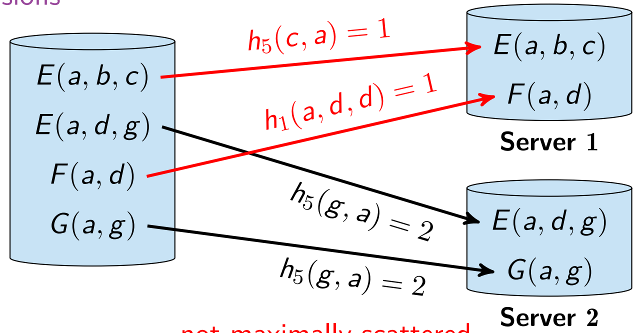


Proof Ingredient: Scattering Hash Functions

Two facts meet on the same server, if and only if,

- they were hashed by the same hash function, and
- according to the same values.

No "accidental" collisions



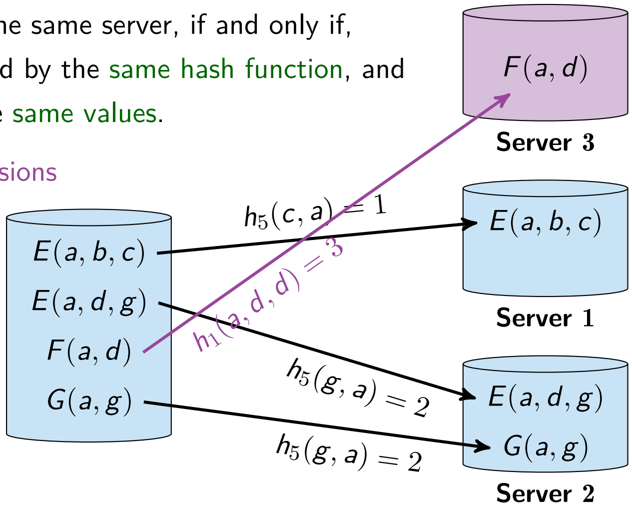
not maximally scattered

Proof Ingredient: Scattering Hash Functions

Two facts meet on the same server, if and only if,

- they were hashed by the same hash function, and
- according to the same values.

No "accidental" collisions

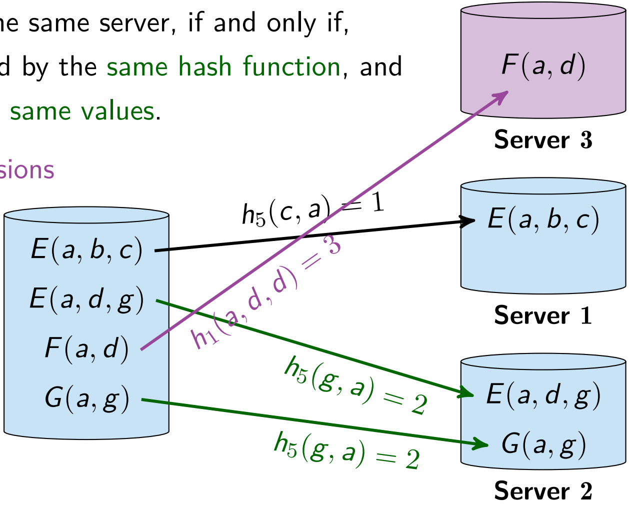


Proof Ingredient: Scattering Hash Functions

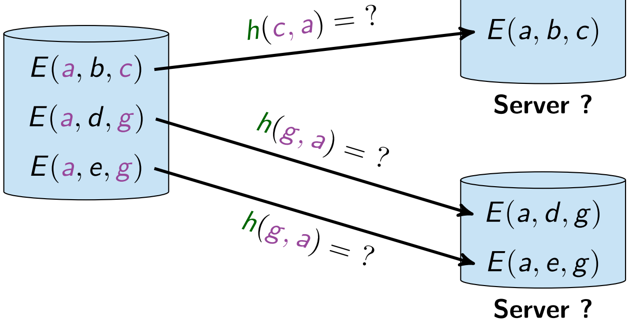
Two facts meet on the same server, if and only if,

- they were hashed by the same hash function, and
- according to the same values.

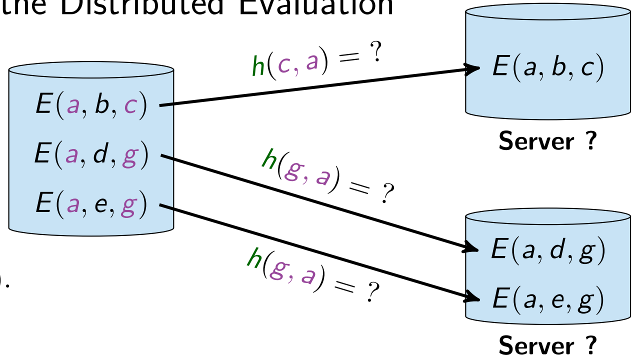
No “accidental” collisions



Proof Ingredient: Simulating the Distributed Evaluation



Proof Ingredient: Simulating the Distributed Evaluation

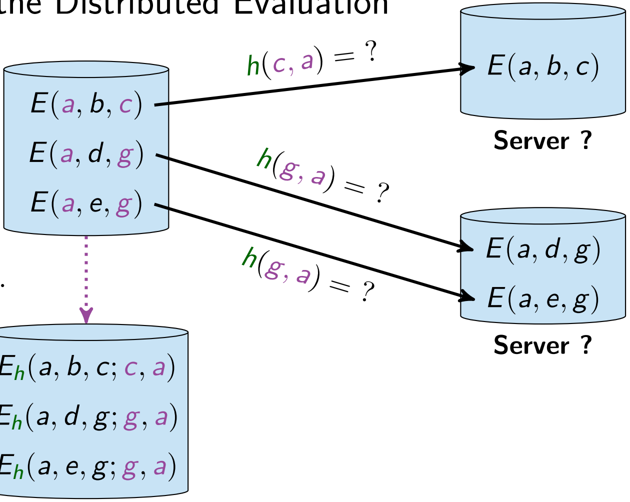


$$E_h(x, y, z; z, x) \leftarrow E(x, y, z).$$

- E_h is new relation symbol
- z, x store hash values

Proof Ingredient: Simulating the Distributed Evaluation

$$E_h(x, y, z; z, x) \leftarrow E(x, y, z).$$



- E_h is new relation symbol
- z, x store hash values

Proof Ingredient: Alternating Tree Automata

Test containment of input program and compiled program

- The compiled program has exponential size
- The containment problem is 2ExpTime-complete (Bourhis, Krötzsch, and Rudolph 2015) involved proof, uses alternating tree automata

Proof Ingredient: Alternating Tree Automata

Test containment of input program and compiled program

- The compiled program has exponential size

  … but each rule has polynomial size

- The containment problem is 2ExpTime-complete
  (Bourhis, Krötzsch, and Rudolph 2015)
  involved proof, uses alternating tree automata

  … but only doubly exponential in the size of rules

  requires very technical analysis

Parallel-Correctness

Theorem

In the non-transitive setting, parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*

is 2ExpTime-*complete.*

Parallel-Correctness

Theorem

In the non-transitive setting, parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*

is 2ExpTime*-complete.*

Corollary

In the non-transitive setting, parallel-correctness for monadic Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*

is 2ExpTime*-complete.*

Parallel-Correctness

Theorem

In the non-transitive setting, parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*

is 2ExpTime-complete.

C[illegible]llary

~~*In the non-transitive s[illegible]llel-correctness for monadic D[illegible],*~~

- ~~*hash policy schemes, and*~~
- ~~*data-movi[illegible]ution constraints*~~

~~*[illegible] 2ExpTime-complete.*~~

Parallel-Correctness

Theorem

In the non-transitive setting, parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*

is 2ExpTime-complete.

Theorem

In the non-transitive setting, parallel-correctness for monadic Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*

is undecidable.

Conclusion

Theorem

Parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*
- *with the polynomial communication property*

is 2ExpTime-*complete.*

Two settings with polynomial communication property:

❶ modest data-moving distribution constraints

❷ non-transitive setting

Conclusion

Theorem

Parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*
- *with the polynomial communication property*

is 2ExpTime-complete.

Two settings with polynomial communication property:

1. modest data-moving distribution constraints
2. non-transitive setting

Further Prospects

- Using hash policy schemes for communication also yields 2ExpTime-completeness
- Parallel-boundedness (please, ask question)

Conclusion

Theorem

Parallel-correctness for frontier-guarded Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*
- *with the polynomial communication property*

is 2ExpTime-complete.

Two settings with polynomial communication property:

1. modest data-moving distribution constraints
2. non-transitive setting

Further Prospects

- Using hash policy schemes for communication also yields 2ExpTime-completeness
- Parallel-boundedness (please, ask question)

Open Questions

- Boundary of decidability for parallel-correctness?
- Other formalism to specify policies?

Parallel-Boundedness

Parallel-boundedness

There is a bound $r \in \mathbb{N}$ such that for every database in the distributed evaluation after r communication rounds no new facts are computed.

- Local computations may be unbounded!
- Does not imply FO-definability

Classical boundedness

There is a bound $k \in \mathbb{N}$ such that for every database the fixed point computation terminates after k iterations.

- Classical boundedness implies FO-definability

Parallel-Boundedness

Parallel-boundedness

There is a bound $r \in \mathbb{N}$ such that for every database in the distributed evaluation after r communication rounds no new facts are computed.

Classical boundedness

There is a bound $k \in \mathbb{N}$ such that for every database the fixed point computation terminates after k iterations.

- Local computations may be unbounded!
- Does not imply FO-definability

- Classical boundedness implies FO-definability

Theorem

Parallel-boundedness for frontier-guarded Datalog programs,

- *hash policy schemes, and*
- *data-moving distribution constraints with the polynomial communication property*

that are parallel-correct is 2ExpTime-*complete.*

Simulation of the Distributed Evaluation

Rules for local computation

$$T(y) \leftarrow F(x, y), R(x).$$

Simulation of the Distributed Evaluation

Rules for local computation

$$T(y) \leftarrow F(x, y), R(x). \xrightarrow[\text{and variables}]{\text{add hash index}} T_h(y; u, v) \leftarrow F_h(x, y; u, v), R_h(x; u, v).$$

Simulation of the Distributed Evaluation

Rules for local computation

$$T(y) \leftarrow F(x,y), R(x). \xrightarrow[\text{and variables}]{\text{add hash index}} T_h(y; u, v) \leftarrow F_h(x,y; u, v), R_h(x; u, v).$$

Rules for communication

$$T_h(y; u, v) \leftarrow F_h(x,y; u, v), \mathbf{R_h}(\mathbf{x}; \mathbf{u}, \mathbf{v}).$$

$$S(z)@\kappa, R(x)@\lambda \rightarrow \mathbf{R}(\mathbf{x})@\kappa$$

Simulation of the Distributed Evaluation

Rules for local computation

$$T(y) \leftarrow F(x,y), R(x). \xrightarrow[\text{and variables}]{\text{add hash index}} T_h(y; u, v) \leftarrow F_h(x, y; u, v), R_h(x; u, v).$$

Rules for communication

$$T_h(y; u, v) \leftarrow F_h(x, y; u, v), \mathbf{R_h}(\mathbf{x}; \mathbf{u}, \mathbf{v}). \quad S(z)@\kappa, R(x)@\lambda \rightarrow \mathbf{R}(\mathbf{x})@\kappa \xrightarrow{\text{inline}} T_h(y; u, v) \leftarrow F_h(x, y; u, v), S_h(z; u, v), R_{h'}(x; w, t).$$

Simulation of the Distributed Evaluation

Rules for local computation

$$T(y) \leftarrow F(x,y), R(x). \quad \xrightarrow[\text{and variables}]{\text{add hash index}} \quad T_h(y;u,v) \leftarrow F_h(x,y;u,v), R_h(x;u,v).$$

Rules for communication

$$\begin{array}{c} T_h(y;u,v) \leftarrow F_h(x,y;u,v), \mathbf{R_h}(\mathbf{x};\mathbf{u},\mathbf{v}). \\ S(z)@\kappa, R(x)@\lambda \rightarrow \mathbf{R}(\mathbf{x})@\kappa \end{array} \quad \xrightarrow{\text{inline}} \quad \begin{array}{c} T_h(y;u,v) \leftarrow F_h(x,y;u,v), \\ S_h(z;u,v), R_{h'}(x;w,t). \end{array}$$

For all combinations of

- hash functions and variables,
- distribution constraints **recursively** (bounded by polynomial communication property)

Parallel-Correctness for Monadic Datalog

Monadic Datalog: Head atoms are unary

$$R(y) \leftarrow S(y), F(w).$$
$$S(y) \leftarrow E(x, y, z).$$

Parallel-Correctness for Monadic Datalog

Monadic Datalog: Head atoms are unary

$$R(y) \leftarrow S(y), F(w).$$
$$S(y) \leftarrow E(x, y, z).$$

rewrite

$$R(y) \leftarrow S(y), F(w), \mathbf{E}(\mathbf{x}, \mathbf{y}, \mathbf{z}).$$
$$S(y) \leftarrow E(x, y, z).$$

frontier-guarded

Parallel-Correctness for Monadic Datalog

Monadic Datalog: Head atoms are unary

$$R(y) \leftarrow S(y), F(w). \qquad \xrightarrow{\text{rewrite}} \qquad R(y) \leftarrow S(y), F(w), \mathbf{E}(\mathbf{x}, \mathbf{y}, \mathbf{z}).$$
$$S(y) \leftarrow E(x, y, z). \qquad\qquad\qquad S(y) \leftarrow E(x, y, z).$$

frontier-guarded

Works in classical setting but does not preserve parallel-correctness!

Parallel-Correctness for Monadic Datalog

Monadic Datalog: Head atoms are unary

$$R(y) \leftarrow S(y), F(w).$$
$$S(y) \leftarrow E(x, y, z).$$

rewrite →

$$R(y) \leftarrow S(y), F(w), \mathbf{E}(\mathbf{x}, \mathbf{y}, \mathbf{z}).$$
$$S(y) \leftarrow E(x, y, z).$$

frontier-guarded

Works in classical setting but does not preserve parallel-correctness!

Theorem

In the non-transitive setting, parallel-correctness for monadic Datalog,

- *hash policy schemes, and*
- *data-moving distribution constraints*

is undecidable.

Literature

Bourhis, Pierre, Markus Krötzsch, and Sebastian Rudolph (2015). "Reasonable Highly Expressive Query Languages." In: International Joint Conference on Artificial Intelligence, IJCAI 2015, pp. 2826–2832.

Ketsman, Bas, Aws Albarghouthi, and Paraschos Koutris (2018). "Distribution Policies for Datalog." In: International Conference on Database Theory, ICDT 2018, 17:1–17:22.