SIHJoin: Querying Remote and Local Linked Data

Günter Ladwig, Thanh Tran
Extended Semantic Web Conference 2011, Heraklion

Institute of Applied Informatics and Formal Description Methods (AIFB)

Contents

- Motivation
  - Linked Data Query Processing
  - Challenges
- Stream-based Linked Data Query Processing
  - Symmetric Index Hash Join
  - Local data access
  - Cost model
- Evaluation
- Conclusion

MOTIVATION

What is Linked Data Query Processing?

- Linked Data
  - RDF data that is distributed among a high number of sources
  - Sources can be retrieved by **dereferencing** their URI
  - Sources contain **links** to other sources
- Linked Data Query Processing [Hartig et al., 2009]

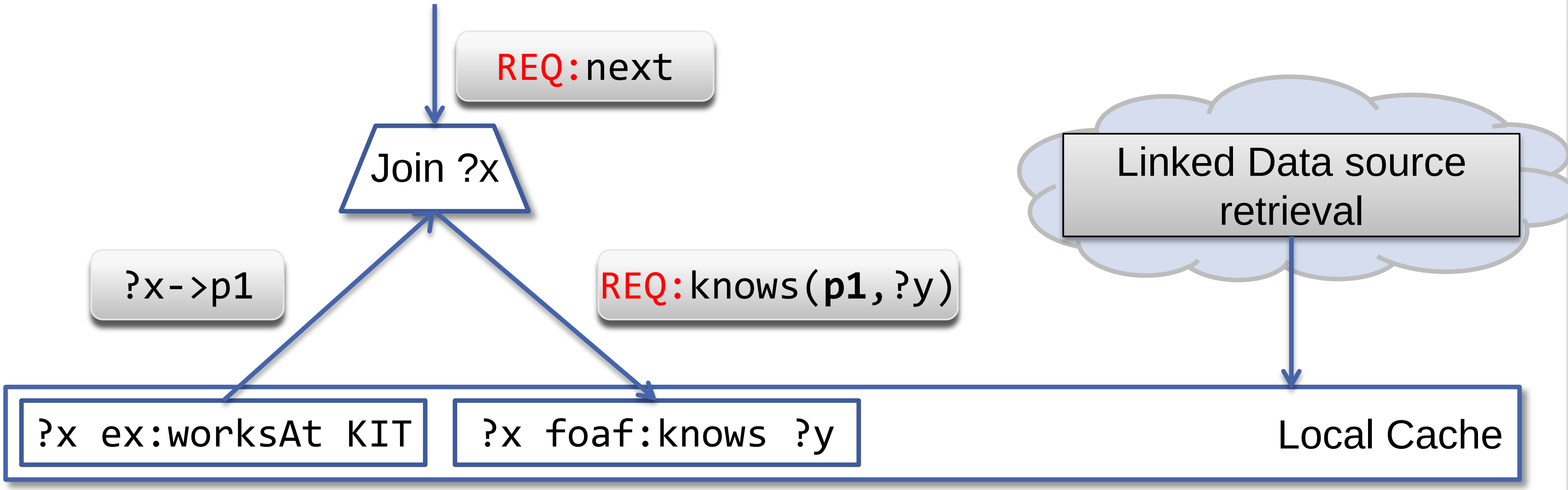
```
SELECT ?x ?n WHERE {
    ?x ex:worksAt dbpedia:KIT .
    ?x foaf:knows ?y .
    ?y foaf:name ?n .
}
```

  - Answer queries by retrieving Linked Data sources, instead of using a triple store or SPARQL endpoint
  - Data is retrieved during query processing via **live lookups**
  - We employ exploration-based link traversal [Hartig et al., 2009]

Challenges

- Large number of sources (hundreds!)
- Unpredictable network latency
- Sources discovered at runtime
- What about locally stored data?

Pull/Iterator-based Join Operation



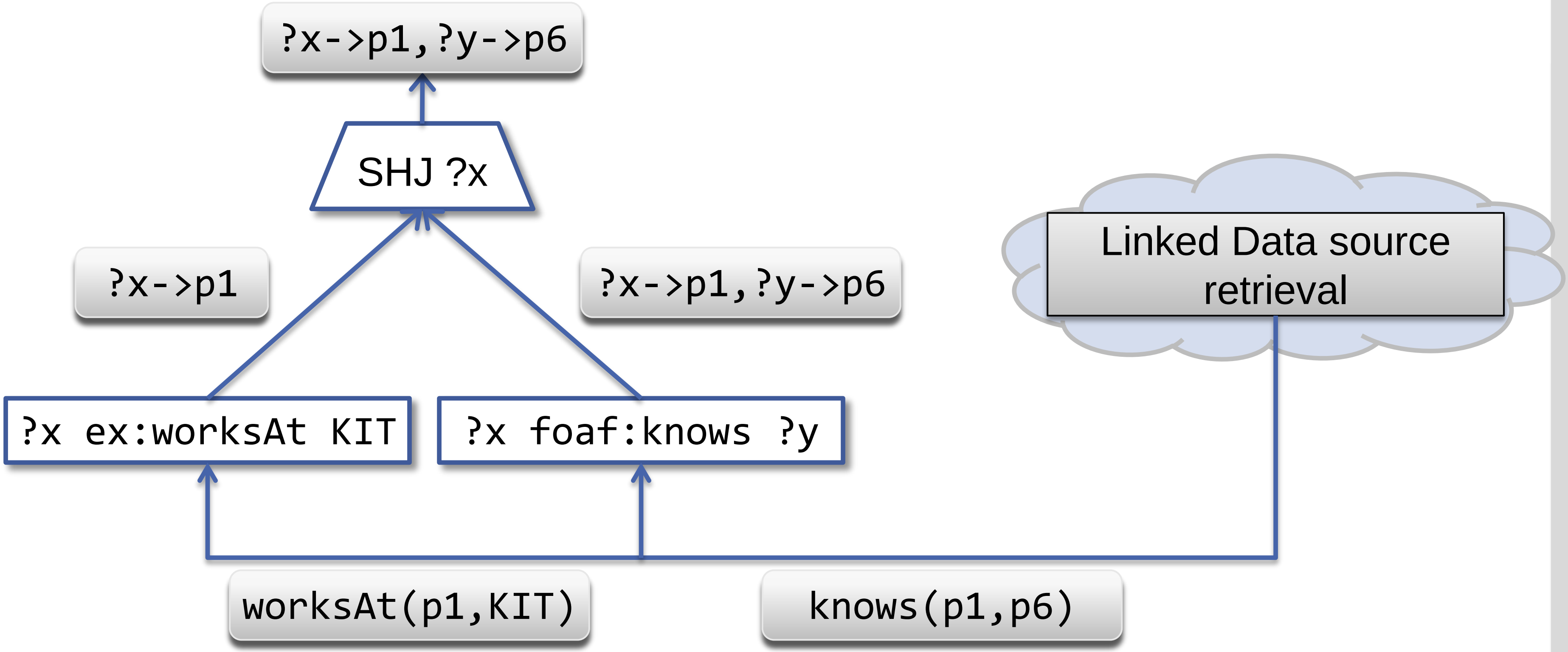
- What if data for a request has not been retrieved yet?
  - Whole query plan is **blocked**, even if there might be data available for other parts
- Can we be sure that matching data is complete?
  - No! Yet to be retrieved sources may contain additional data that is not “seen” by the join

STREAM-BASED LINKED DATA QUERY PROCESSING

Stream-based Linked Data Query Processing

- We treat incoming data as finite streams
- Join operator based on Symmetric Hash Join
  - Well known from database research
  - Uses hash tables to keep track of seen inputs
- Push-based mode
  - Execution is driven by the incoming data, not the results
  - Non-blocking
  - Data can arrive on any input and in any order

Push-based Symmetric Hash Join



- Where is the local data?

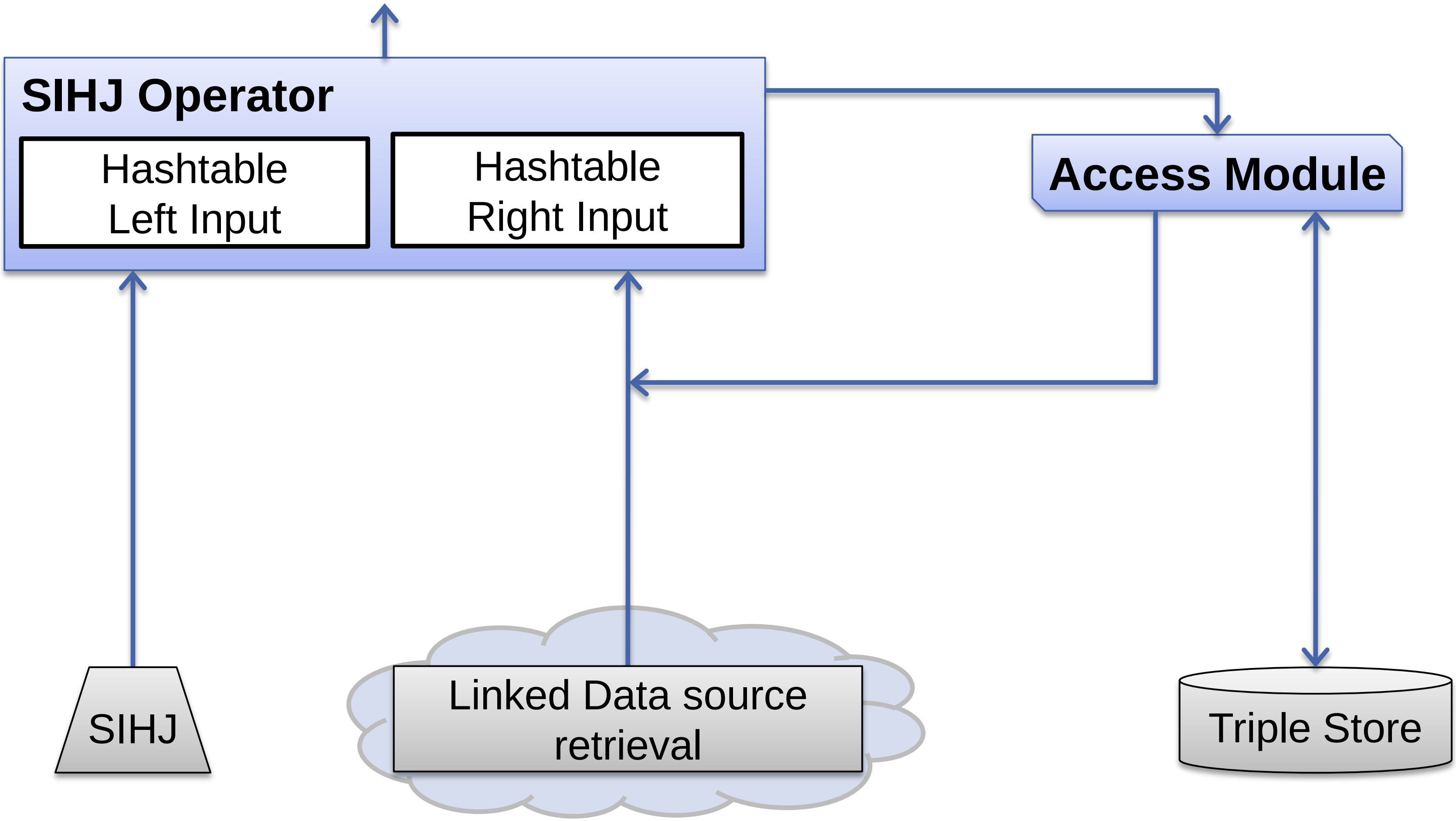
Querying Remote and Local Linked Data

- Linked Data Query Processing was concerned with remote data only
- How do we execute queries over both, remote data and locally stored data?
  - For example: private data that is linked to publicly available data
- Conceptually local data can be seen as yet another data source
  - Naïve solution: treat locally stored data as a remote source and process them in the same way. Inefficient!
- We can do better
  - Local data is usually stored in triple stores and accessible using special indexes, use them!

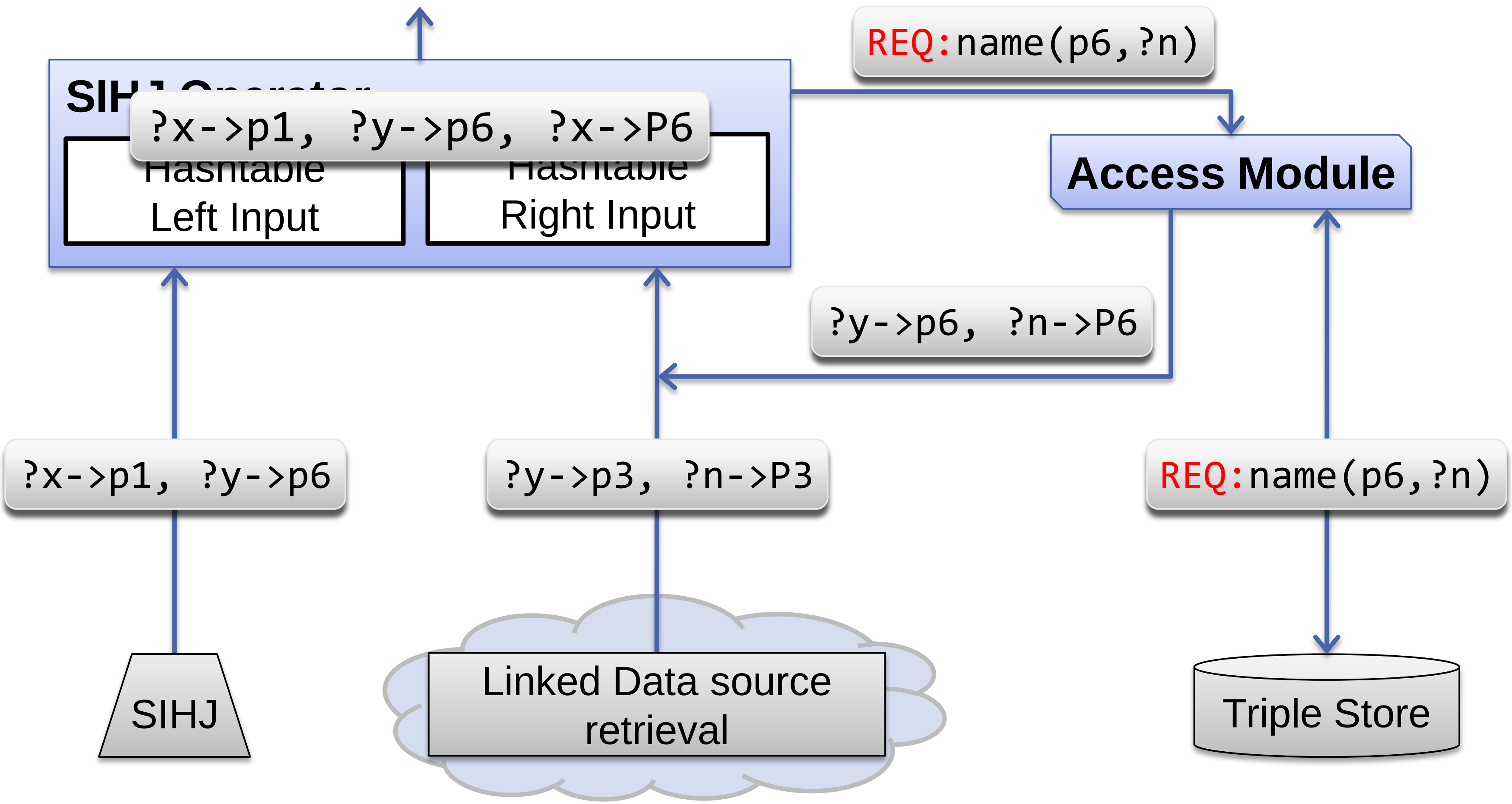
Symmetric Index Hash Join

- Extends push-based SHJ with pull-based **local index access**
- Load only data that is guaranteed to produce join results
  - Instantiate triple patterns with binding values
- Preserve the push-based operation to avoid blocking
- **Access modules** encapsulate access to local data and indexes
  - One for each SIHJ operator
  - Loads data **asynchronously**
  - Pushes data back into SIHJ operator

Symmetric Index Hash Join



Symmetric Index Hash Join



SIHJ Cost Model

- We define a cost model for the SIHJ operator
  - Costs in terms of tuples accessed and cost of the physical operations needed
  - Abstract from concrete implementations and (some) data structures
- Cost of a SIHJ between two inputs *A* and *B*
  - Cost of joining tuples arriving on the left input
  - Cost of joining tuples arriving on the right input
  - Cost of the access module

$$C_{A \bowtie B} = C_{A \rtimes B} + C_{A \ltimes B} + C_{AM}$$

SIHJ Cost Model

- Operations carried out for each tuple on left input
  - Insert into hash table for A, weight factor: I_h
  - Probe hash table for B, weight factor: P_h
  - Create join results, weight factor: J, size of results given by join selectivity φ
  - Send request to access module, weight factor: R

$$C_{A \bowtie B} = |A| \left(I_h + P_h + \varphi \cdot |B| \cdot \frac{|A|}{|A| + |B|} \cdot J + R \right)$$

- To use cost model for query optimization, weight factors have to obtained through experiments

Completeness

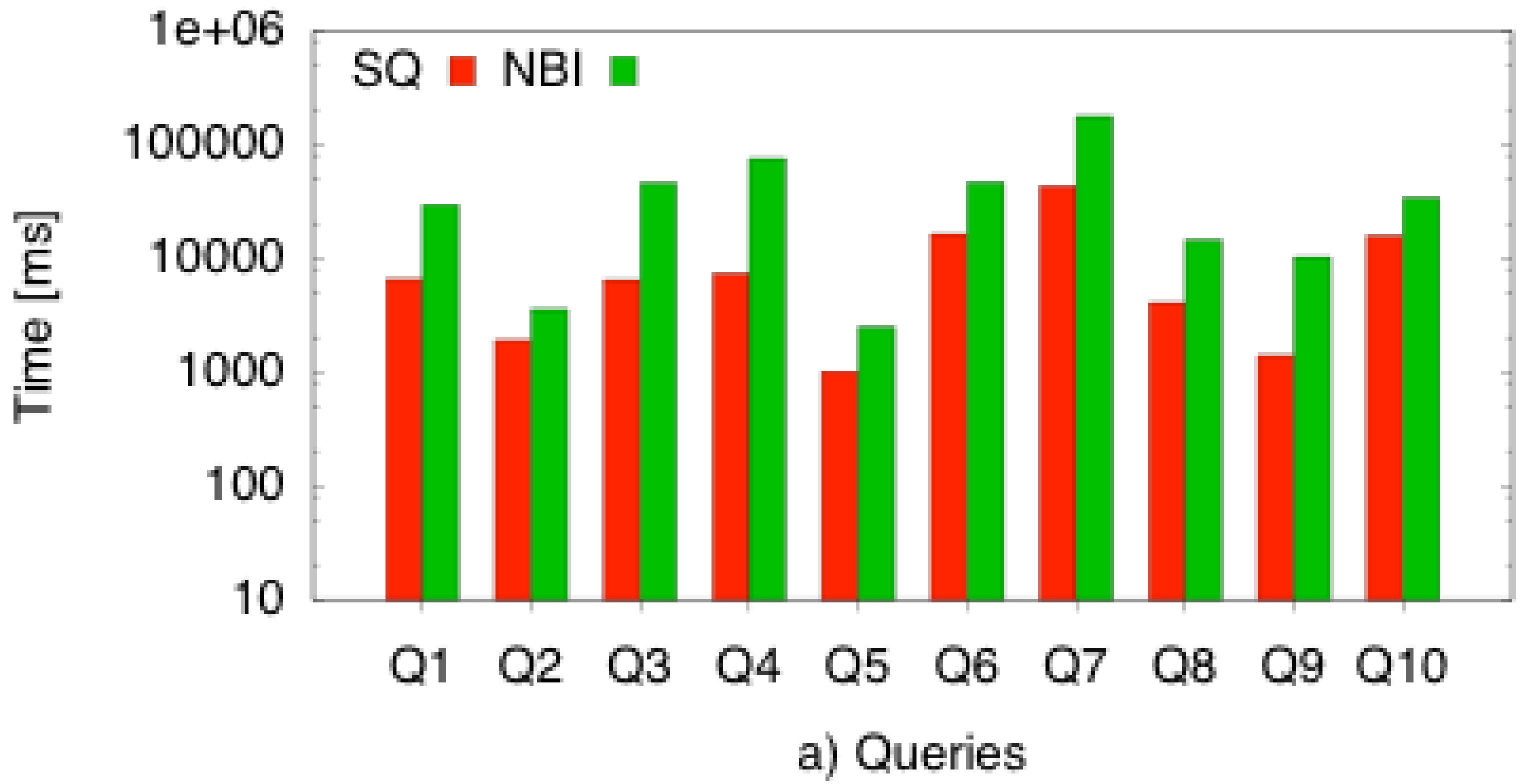
- Non-Blocking Iterator [Hartig et al., 2009]
  - Iterator model (pull-based)
  - Assumption: given a bound triple pattern *t*, all matching triples are contained in sources mentioned in *t*
  - While this is often the case, Linked Data sources may contain arbitrary data
  - Completeness **is not guaranteed**
- SIHJ guarantees completeness
  - Operation is completely symmetrical: when a binding arrives on either input, it is always inserted into the corresponding hash table
  - SIHJ can produce all join results, because it **keeps track of all previously seen inputs**
  - Final result is always complete with respect to retrieved sources
  - However: memory overhead

EVALUATION

Evaluation – Overall Performance

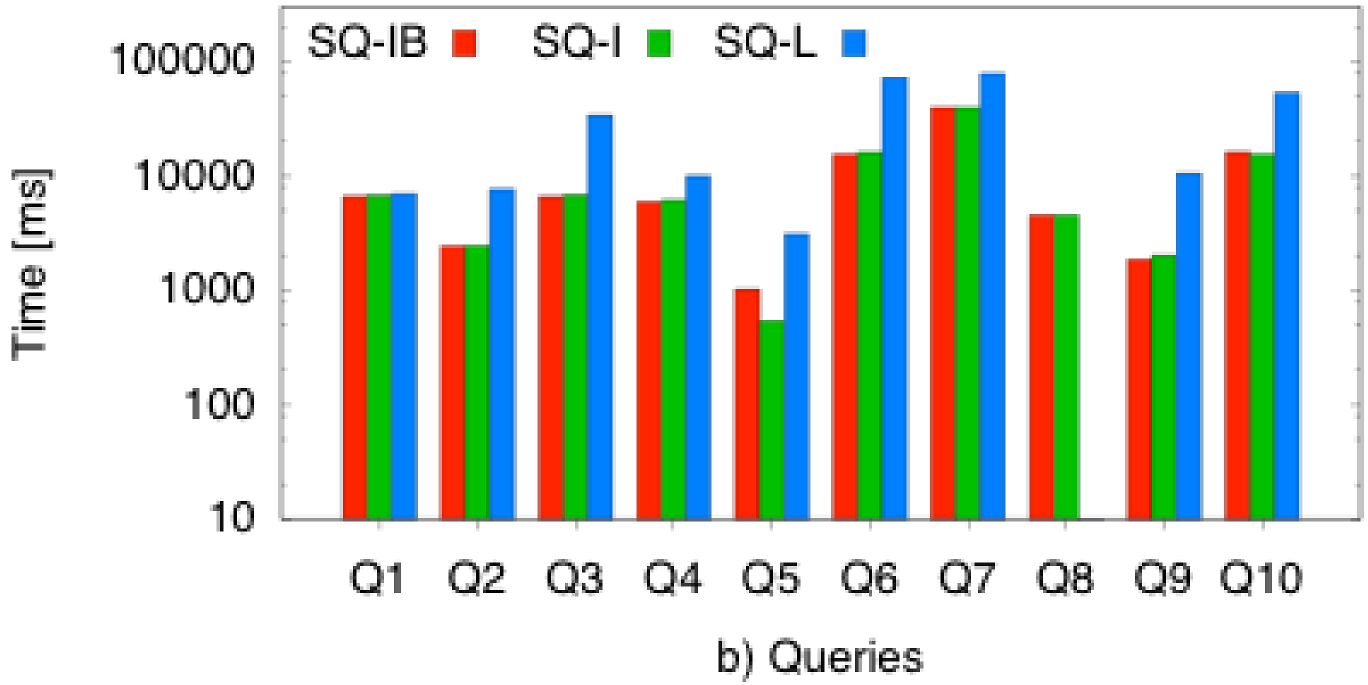
- Show benefits of stream-based query processing
- Systems
  - Remote data only: SQ (pushed-based SHJ) and NBI [Hartig et al. 2009] (SQUIN implementation of NBIJ)
  - Remote and local data: SQ-L (naïve solution), SQ-I (uses local indexes) and SQ-IB (local indexes and batching)
- Dataset
  - Several popular datasets, such as DBpedia, NYT, etc.: 166M triples
  - Dataset was split into remote (90%) and local (10%) data
  - Remote data deployed on CumulusRDF Linked Data server
  - Local data indexed in triple store
- Queries
  - 10 BGP queries of different complexity w.r.t query size and number of sources retrieved

Results – Overall Performance



- Averages
  - SQ: 9699.18ms
  - NBI: 41704.27ms
- Improvement of 77%

Results – Overall Performance

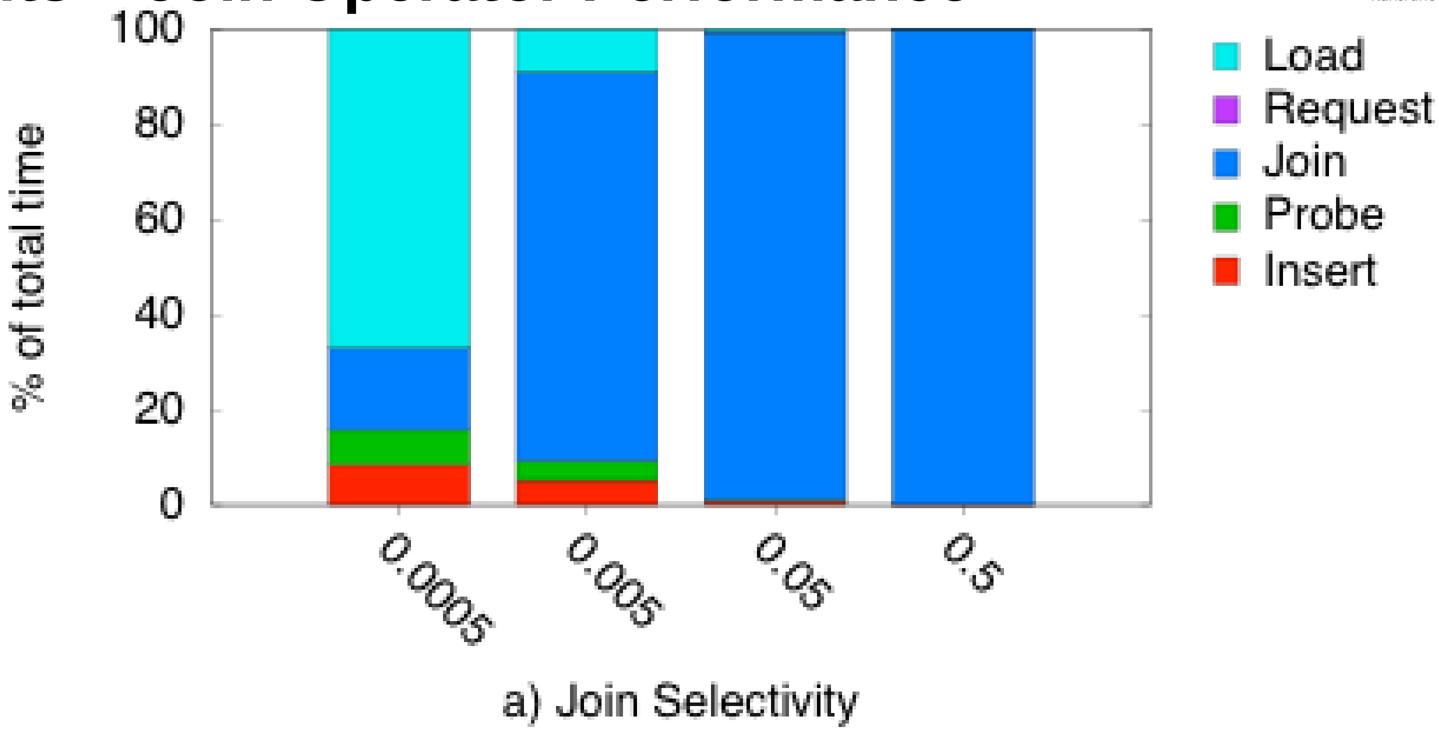


- Averages
  - SQ-IB: 9366.39ms
  - SQ-I: 9396.18ms
  - SQ-L: 28448.98ms
- Improvement of 67% of SQ-IB over SQ-L

Evaluation – Join Operator Performance

- Focus on join processing with SIHJ and NBIJ
- Systems
  - SQ-L, SQ-I, SQ-IB
  - NBI (custom implementation)
- Several **synthetic** datasets with different data characteristics
  - a,b: size of left and right input
  - ρ: fraction of right input that is local data
  - φ: join selectivity
  - s: number of sources for remote part of right input
- Remote data stored in-memory, local data on disk

Results – Join Operator Performance



- b = 10000, ρ = 0.2, s = 200
- With higher selectivity values, join creation dominates processing time

$$C_{A \bowtie_{SIHJ} B} = |A|(I_h + P_h) + |B|(I_h + P_h) + \boxed{\cdot \varphi |A||B| \cdot J}$$

Conclusion

Karlsruhe Institute of Technology

- Proposed Symmetric Index Hash Join for Linked Data Query Processing
  - Integrates push- and pull-based mechanisms
  - Supports remote and local Linked Data
- Cost models for SIHJ and previously proposed NBI
- SIHJ guarantees completeness (w.r.t. to retrieved sources)
- Evaluation
  - Total query time improved by 77% over NBI on average
  - Using indexes to integrate local data improves performance by 67% compared to naïve solution