Data and Knowledge Management in Cloud Computing Environment

Ph.D. Presentation

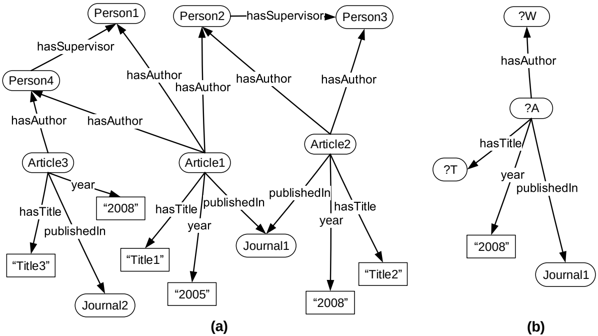
Eleftherios Kalogeros

Advisory Committee Members
Manolis Gergatsoulis, Christos Papathodorou, Timos Sellis

Department of Archives, Library Science and Museology
Ionian University

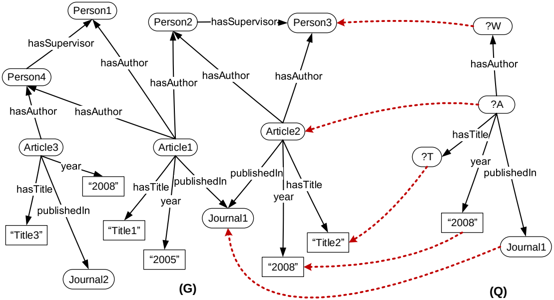
6 July 2022

Data and Query Graphs



(a) a data graph - RDF triples, each triple consist of three values - subject value URI, predicate value URI, object value URI or Literal
(b) a query graph - BGP SPARQL query, each query triple contains also three values - each value may be a variable (?W)

Embeddings (Answer)



Query (Q): Find the articles (variable ?A), its authors (variable ?W) and titles (variable ?T) published in Journal1 at year "2008"

Two embeddings (Answers) of the query graph Q in the data graph G (1) (?A, ?W, ?T) = (Article2, Person3, "Title2")

(2) (?A, ?W, ?T) = (Article2, Person2, "Title2")

Ph.D. Thesis Study

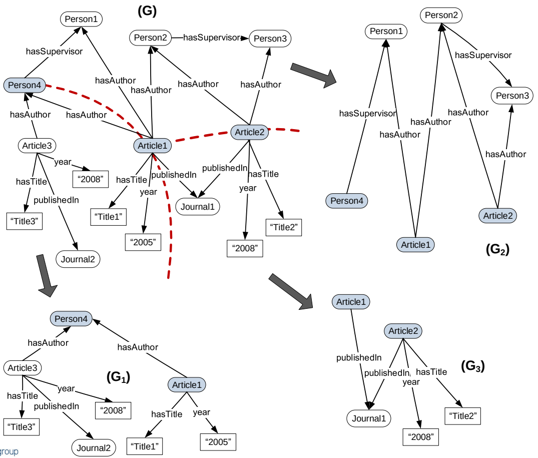
In all our approaches we study the problem of distributed processing (BGP SPARQL queries) over linked data (RDF).

Our methodology is

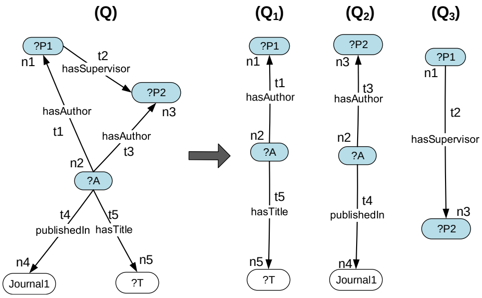
(1) Partition data graph into graph segments and store them into different nodes of a cluster of machines
(2) Decompose query (query graph) into subqueries
(3) Process each subquery on each data fragment in parallel
(4) Combine the intermediate results in parallel to compute the answers of the initial query.

Data Decomposition

The dark nodes correspond to the **border nodes** (common nodes) between the data graph segments. ($\mathcal{B}(G_1) = \{Person4, Article1\}$)
Notice that the graph decomposition appearing is **non-redundant**



Query Graph Decomposition

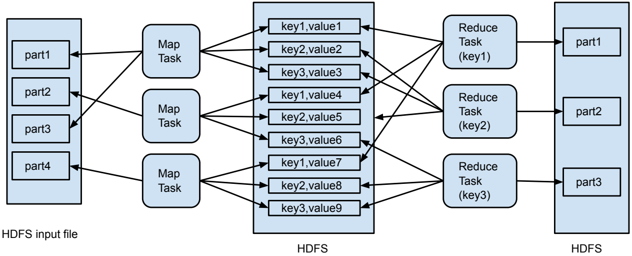


A (**non-redundant**) decomposition $\mathcal{D}_Q$ of a query Q into 3 subqueries. The **border nodes** (common nodes) between the query graph segments are: $\mathcal{B}(Q_1) = \{n1, n2\}$, $\mathcal{B}(Q_2) = \{n2, n3\}$, $\mathcal{B}(Q_3) = \{n1, n3\}$.

Missing Border Nodes

MBN = [(n1, Q2), (n2, Q3), (n3, Q1)].

Apache Hadoop (MapReduce)



MapReduce Programming Model

Apache Spark and NoSQL(MongoDB)

Apache Spark is **in-memory** fault tolerance computation engine.
(1) Spark **RDD** is a collection of elements partitioned across the nodes of the cluster.
(2) Spark **DataFrame** is a table-like abstraction.

MongoDB is a NoSQL document database that manages collections of **JSON** documents

Query evaluation approaches

- Query Evaluation by Joining Partial Embeddings (QEJPE algorithm)
- Query evaluation by decomposing queries into generalized stars (eval-STARS algorithm)
- Query evaluation by data decomposition using replication (QE-with-Redundancy algorithm)
- Query evaluation over data stored in a document database (Doc-based algorithm)

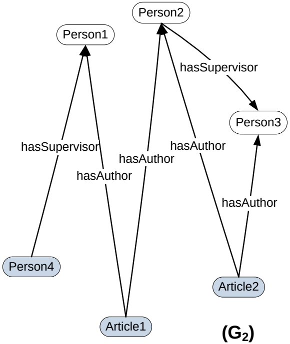
Query Evaluation by Joining Partial Embeddings
(QEJPE algorithm)

QEJPE - Partial embeddings

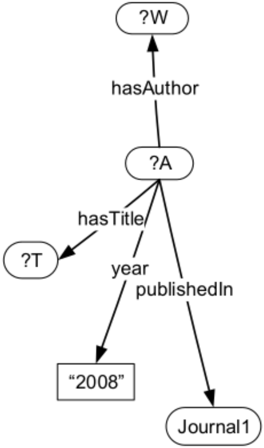
Query Q: (?W,?A,?T)

Answer:(Article2,Person2,"Title2"),
(Article2,Person3,"Title2")

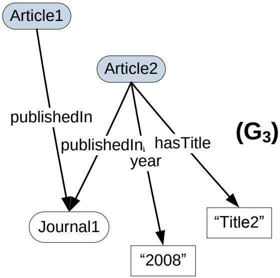
Partial Embeddings:
(Article2,Person2,*),
(Article2,Person3,*)



(G_2)



Partial Embeddings:(Article2,*,"Title2)



(G_3)

QEJPE algorithm Strategy

Step 1: Decompose the query Q into a tuple $\mathcal{D}_Q$ of subqueries $Q_1, \ldots, Q_n$, with $n \geq 1$.

Step 2: Compute all possible useful partial embeddings of each subquery Q_j over each data graph segment G_i of G.

Step 3: For each subquery Q_j, collect all the partial embeddings of Q_j obtained in Step 2 and join them to get total embeddings of Q_j.

Step 4: To construct the total embeddings (i.e. answers) of Q, join the total embeddings obtained in Step 3 by using one embedding for each subquery.

DBIS database & information systems group ionian university

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ · ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ 1984

The Preprocessing Phase

- Q is decomposed into a set of subqueries $Q_1, \ldots, Q_n$.
- Nodes are numbered: Border nodes (1 to $|B(Q)|$) and non-border nodes ($|B(Q)| + 1$ to $|nd(Q)|$), of Q.
- The set $MBN = \{(b_i,\ Q_j)|\ b_i \in BN$ and $b_i \notin nd(Q_j)\}$, is constructed.
- The embeddings of a (sub)query are represented as triples of tuples $(BNt, NBNt, tF)$, where:
  - BNt (resp. $NBNt$) stores the images of border (resp. non-border) nodes.
  - Asterisks ('*') are used to represent missing values.
  - tF keeps tracks for the triples participating in the embedding ('+'/'-' sign in the corresponding place of tF).

DBIS
database & information systems group
ionian university

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ · ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ 1984 ·

QEJPE Algorithm Based on Map-Reduce

	Mapper1	Reducer1	Mapper2	Reducer2
input **key**,*values*	**Q_i**,*G_j*	**Q_i**, *useful embeddings*	**Q_i**, *full embeddings* **Q_i**, *missing border node values*	**bn′**, *(Q_i,nbn)*
process	Computes **useful** (total or partial) **embeddings** of Q_i in G_j	Computes the **total embeddings** of Q_i in G based on useful embeddings of Q_i in segments G_1, ..., G_m	Gets the embeddings of a Q_i and **fills in its missing border node** values using values from the embeddings of other subqueries Q_j.	Selects one embedding for each subquery in (Q_1, . . . , Q_n) and joins them to construct an answer of Q (Note: the joined embeddings are, by construction, compatible)
output **key**,*values*	**Q_i**,*useful embeddings*	**Q_i**, *full embeddings* **Q_j**, *missing border node values (bni,value)*	**bn′**, *(Q_i,nbn)*	Answers of Q in G

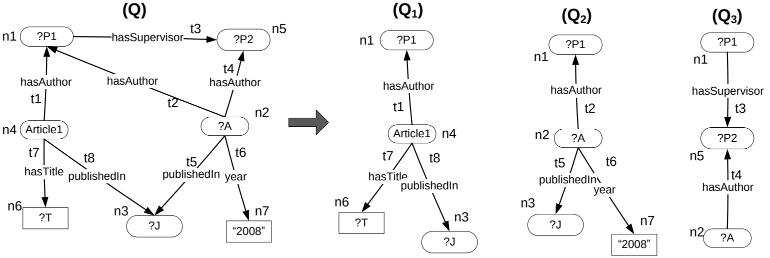
QEJPE algorithm

DBIS
database & information systems group
ionian university

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ · ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ 1984

Query evaluation by decomposing queries into generalized stars (eval-STARS algorithm)

Generalized star queries



A query Q is called a generalized star query if there exists a node c, called the central node of Q such that c is subject or object in all query triples. (i.e. n4 node with value Article1 in Q1).

A query Q that consists from one query triple is a generalized star query. That means that every Q can be decomposed into a set of generalized star subqueries

eval-STARS algorithm Strategy

Step 1: Decompose the query Q into a tuple of generalized star subqueries $\mathcal{D}_Q = (Q_1, \ldots, Q_n)$, with $n \geq 1$.

Step 2: Compute all possible embeddings of each triple in Q over each data graph segment G_i of G.

Step 3: For each subquery Q_j, collect the embeddings of all the triples in Q_j and join compatible embeddings in all possible ways to compute the total embeddings of Q_j in G.

Step 4: To construct the total embeddings (i.e. answers) of Q, join the total embeddings obtained in Step 3 by using one embedding for each subquery.

DBIS database & information systems group ionian university

eval-STARS Algorithm Based on Map-Reduce

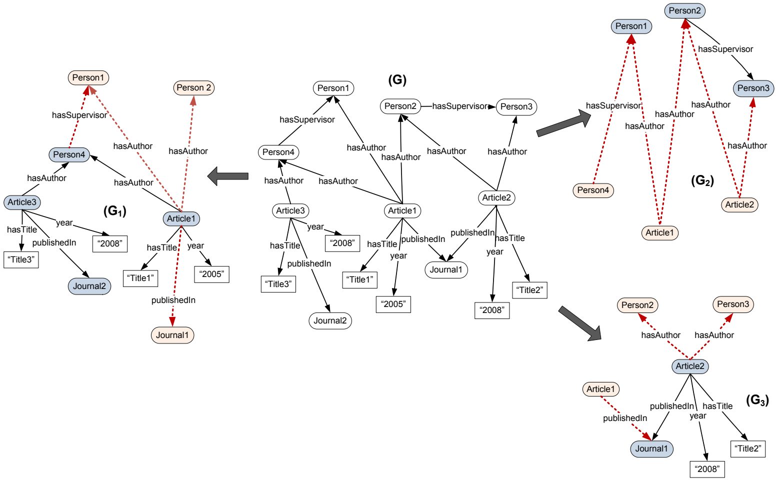
	Mapper1	Reducer1	Mapper2	Reducer2
input **key**,*values*	Q_i,G_j	(Q_i,v),(n_k,u)	Q_i,*full embeddings* Q_i, *missing border node values*	**bn'**, (Q_i,nbn)
process	Computes all the embeddings **of each triple** of Q_i in G_j that map the central node c_i to a **border node** Computes all the embeddings of Q_i in G_j, which map c_i to a **non-border node** of G_j	Computes all the embeddings of Q_i that map central node c_i of Q_i to v, combining (cartesian product) all the values u from all the nodes n_k	Gets the embeddings of a Q_i and **fills in its missing** border **node values** using values from the embeddings of other subqueries Q_j.	Selects one embedding for each subquery in (Q_1, . . . , Q_n) and joins them to construct an answer of Q (Note: the joined embeddings are, by construction, compatible)
output **key**,*values*	[(Q_i, e(c_i)),(o, e(o))] or [(Q_i , e(c_i)),(s, e(s))] Q_i, *full embeddings* Q_j, *missing border node values* (bn_i,value)	Q_i,*full embeddings* Qj, *missing border node values* (bni,value)	**bn'**, (Qi,nbn)	Answers of Q in G

eval-STARS algorithm

DBIS
database & information systems group
ionian university

Query evaluation by data decomposition using replication
(QE-with-Redundancy algorithm)

Star-oriented decomposition



QE-with-Redundancy algorithm Strategy

Step 1: Decompose the query Q into a tuple of queries $\mathcal{D}_Q = (Q_1, \ldots, Q_n)$, with $n \geq 1$, such that each query in $\mathcal{D}_Q$ is a subject-object star query.
(A subject-object star query is a generalized star query that has **at least one** query triple whose **subject** is the central node)

Step 2: Compute all possible embeddings of each subquery in $\mathcal{D}_Q$ on every segment in $\mathcal{D}_G$.

Step 3: Compute the embeddings of Q on G by joining compatible embeddings of the subqueries $Q_1, \ldots, Q_n$.

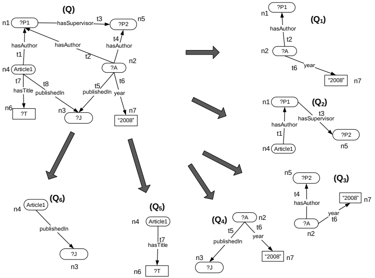
DBIS
database & information systems group
ionian university

QE-with-Redundancy Algorithm Based on Map-Reduce

	Mapper1	Reducer1	Mapper2	Reducer2
input **key**,*values*	Q_i,G_j		**Q_i,e(CB(Q))**, *full embeddings* **Q_i,e(CB(Q))**, *missing border node values (bn_j,value)*	**bn′**, *(Q_j,nbn)*
process	Computes all the embeddings of Qi in Gj		Gets the embeddings of a Q_i and fills **in its missing border node values** using values from the embeddings of other subqueries Q_j.	Selects one embedding for each subquery in (Q_1, . . . , Q_n) and joins them to construct an answer of Q (Note: the joined embeddings are, by construction, compatible)
output **key**,*values*	**<u>If MBN is not empty</u>** **Q_j,e(CB(Q))**, *full embeddings* **Q_j,e(CB(Q))**, *missing border node values (bn_j,value)* **<u>If MBN is empty</u>** **bn′**, *(Q_j,nbn)*		**bn′**, *(Q_j,nbn)*	**Answers of Q in G**

QE-with-Redundancy algorithm

- **Min-res algorithm** decomposes a query Q into a set of so-subqueries, such that each subquery has at most two variables. It also allows replication of triples that contains at most one variable, and maximizes the number of "constraints" (triples that do not increase the number of variables in the query) in each subquery containing variables. As for the subqueries that do not contain any variable, the algorithm constructs maximal subqueries without redundant constraints.



database & information systems group
ionian university

Query decomposition algorithms (2/3)

- **Max-degree algorithm** decomposes a query Q into a set of so-subqueries based on the nodes degrees. We focus on selecting first the subqueries containing as many triples as possible. In each step of the algorithm, it finds the so-query with max degree and removes its edges from the remaining so-stars (i.e. redundancy is not allowed in query decomposition).
- **Max-degree-with-redundancy algorithm** Same as Max-degree algorithm but in each step of the algorithm, edges from the remaining so-stars that add more constraints to the other subqueries do not remove (i.e. redundancy is allowed in query decomposition).

DBIS database & information systems group ionian university

Query decomposition algorithms (3/3)

- In Max-degree algorithm and Max-degree-with-redundancy algorithm in each iteration, if the query resulted by removing the covered triples is not an so-query, then the query is ignored
- **Max-degree-with-reshaping algorithm** In each iteration Q if the query resulted by removing the covered triples is not an so-query, we add a query triple (from the covered triples) to the query to construct a new so-query. The selected triple is removed then from the previous selected so-query (new query is also so-query). This approach might reshape the so-queries constructed in the previous iterations

DBIS
database & information systems group
ionian university

MapReduce algorithms - Properties

- QEJPE and eval-STARS algorithms are independent of the way the data graph is decomposed and the way the data graph segments obtained by this decomposition are stored in the nodes of the cluster. QE-with-Redundance algorithm s independent of the choice of the specific partition $\mathcal{N}_{\mathcal{P}}$ of the nodes in $\mathcal{N}(G) - L$
- all algorithms are independent of the algorithm used to compute embeddings
- all algorithms are independent of the specific query decomposition strategy. eval-STARS algorithm is based in generalized star subqueries, QE-with-Redundancy algorithm is based in subject-object star subqueries

DBIS
database & information systems group
ionian university

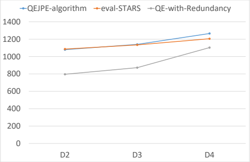
Experiments setup environment

- Cluster with 10 virtual machines with the the following characteristics: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz (8 Cores) with 16GB RAM, 60GB HD, Ubuntu 16.04 LTS, 64-bit Operating System. We used Apache Hadoop v3.1 with HDFS (1 NameNode, 1 Secondary NameMode, 10 DataNodes each one 30GB) and YARN (1 ResourceManager, 10 NodeManagers). The 10 virtual machines were connected through external IP addresses. Python implementation.
- we used four different datasets (D1: 2,731,510 triples - 7 files, D2: 5,486,199 triples - 13 files, D3: 10,979,566 triples - 25 files, D4: 21,961,070 triples - 49 files) in N-Triples format from the Waterloo SPARQL Diversity Test Suite (WatDiv)

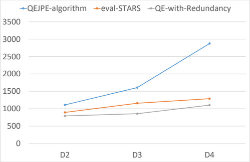
Experimental results (1/4)

Query evaluation in terms of the size of dataset

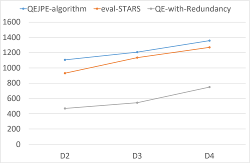
Linear Query



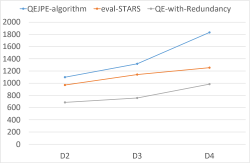
Snowflake Query



Star Query



Average per evaluation algorithm

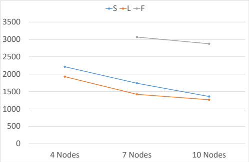


DBIS
database & information systems group
ionian university

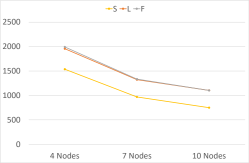
Experimental results (2/4)

Query evaluation in terms of compute nodes size per algorithm

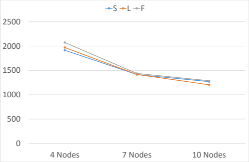
QEJPE-algorithm



QE-with-Redundancy



eval-STARS

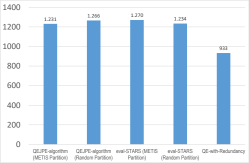


DBIS
database & information systems group
ionian university

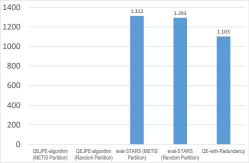
Experimental results (3/4)

Comparison of query evaluation algorithms for a variety of query types

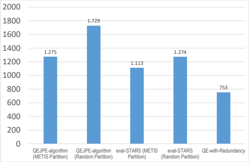
Linear Query Type Evaluation



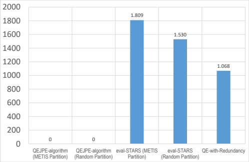
Snowflake Query Type Evaluation



Star Query Type Evaluation



Complex Query Type Evaluation

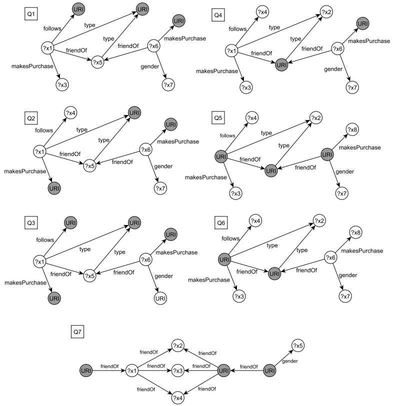


DBIS database & information systems group ionian university

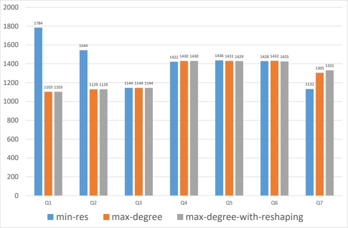
Experimental results (4/4)

Query Decomposition Algorithms Evaluation

Queries



Query Decomposition Algorithms Evaluation

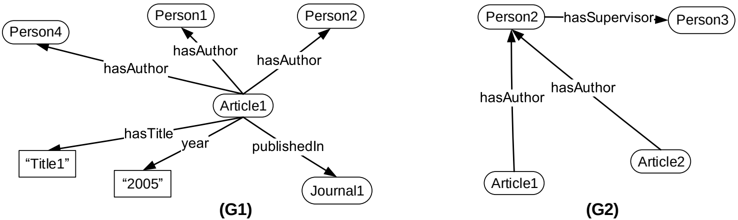


DBIS
database & information systems group
ionian university

Query evaluation approaches - Doc-based Algorithm

Query evaluation over data stored in a document database
(Doc-based algorithm)

Node-partition decomposition



- The six triples that contain the node **Article1** are stored in a **single** JSON file (similar for each node of the data graph)
- The number of JSON files is the number of the nodes
- The answers (embeddings) of each generalized star subquery can be computed in every single JSON file (no JOINS)

Query decomposition algorithm

- Subqueries containing as many as possible non-covered (i.e. non replicated) triples of Q are selected first. In this way, the number of subqueries produced by decomposing the query Q is kept as small as possible. Such selection is based on the observation that as the number of the query triples is increasing the number of results (number or documents matching the subquery) is decreasing.
- Among the subqueries with equal number of non-covered triples the subqueries whose central nodes are URIs precede to our selection, comparing with the subqueries whose central nodes are variables.
- In case that the above criteria are satisfied by more that one generalized star queries we select the query with the maximum number of literals and URI nodes.

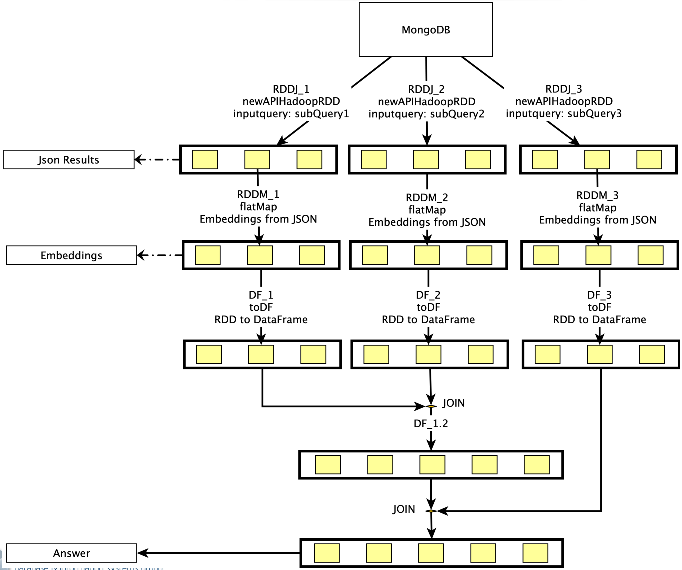
Doc-based algorithm Strategy

- The data graph G is partitioned using the node-oriented partitioning approach.
- The initial query Q is decomposed into a set $\mathcal{D}_Q$ of generalized star subqueries. *(with the constraint that the central node can* **not** *be literal)*.
- We, then, find the embeddings of each subquery in $\mathcal{D}_Q$ on each node-graph segment.
- Finally, we join the compatible embeddings, one for each subquery, in order to construct the embeddings of the query Q.

Apache SPARK and mongoDB implementation (1/2)

- Q is decomposed into a set of generalized star subqueries and each one is transformed into the corresponding MongoDB query.
- Each subquery return (one RDD) the JSON documents that is at least one embedding from the subquery
- RDDs are flattened into relational-like structure. This step is implemented in Spark and no data shuffling between cluster nodes is required (i.e., such a transformation is performed in parallel over each element of the RDD).
- RDDs are translated into Spark DataFrames (optimized join operations)
- DataFrames (one for each subquery) are joined over the common queries nodes/fields in order to compute the answer

Apache SPARK and mongoDB implementation (2/2)



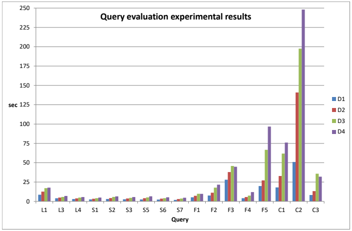
Experiments setup environment

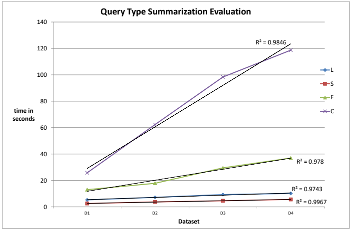
- Cluster with 10 virtual machines with the the following characteristics: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz (8 Cores) with 16GB RAM, 60GB HD, Ubuntu 16.04 LTS, 64-bit Operating System. The 10 virtual machines were connected through external IP addresses. Python implementation.
- Apache Spark (6-node cluster) and MongoDB (1 router server, 1 config server and 5 shards).
- we used four different datasets (D1: 8,773,357 triples, D2: 17,582,410 triples, D3: 26,342,929 triples, D4: 35,112,532 triples) in N-Triples format from the Waterloo SPARQL Diversity Test Suite (WatDiv)

DBIS
database & information systems group
ionian university

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ · ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ 1984

Experimental results (1/2)

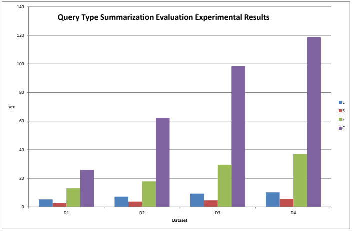


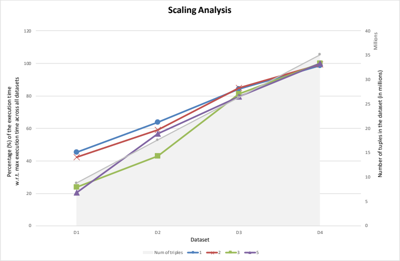


DBIS database & information systems group ionian university

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ · ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ 1984

Experimental results (2/2)





List of Ph.D. publications

- M. Gergatsoulis, C. Nomikos, E. Kalogeros, and M. Damigos, "An Algorithm for Querying Linked Data Using Map-Reduce," in Data Management in Cloud, Grid and P2P Systems - 6th International Conference, Globe 2013, Prague, Czech Republic, August 28-29, 2013, vol. 8059, pp. 51–62
- C. Nomikos, M. Gergatsoulis, E. Kalogeros, and M. Damigos, "A Map-Reduce algorithm for querying linked data based on query decomposition into stars," in Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014, vol. 1133, pp. 224–231
- E. Kalogeros, M. Gergatsoulis, and M. Damigos, "Redundancy in Linked Data Partitioning for Efficient Query Evaluation," in 3rd International Conference on Future Internet of Things and Cloud, FiCloud 2015, Rome, Italy, August 24-26, 2015, pp. 497–504
- E. Kalogeros, M. Gergatsoulis and M. Damigos, "Document based RDF storage method for efficient parallel query processing," in Metadata and Semantic Research - 12th International Conference, MTSR 2018, Limassol, Cyprus, October 23-26, 2018, vol. 846, pp. 13–25
- E. Kalogeros, M. Gergatsoulis and M. Damigos, "Document-based RDF storage method for parallel evaluation of basic graph pattern queries," International Journal of Metadata, Semantics and Ontologies, vol. 14, no. 1, pp. 63–80, 2020
- E. Kalogeros, M. Gergatsoulis, M. Damigos, C. Nomikos, "Efficient query evaluation techniques over large amount of distributed linked data," It will be submitted

DBIS
database & information systems group
ionian university

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ · ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ 1984

Other publications

- M. Damigos, M. Gergatsoulis, and E. Kalogeros, "Distributed evaluation of XPath queries over large integrated XML data," in 18th Panhellenic Conference on Informatics, PCI 2014, Athens, Greece, October 2-4, 2014, 2014, pp. 1–6
- M. Agathos, E. Kalogeros, and S. Kapidakis, "A Case Study of Summarizing and Normalizing the Properties of DBpedia Building Instances," in Research and Advanced Technology for Digital Libraries - 20th International Conference on Theory and Practice of Digital Libraries, TPDL 2016, Hannover, Germany, September 5-9, 2016, vol. 9819, pp. 398–404
- M. Gergatsoulis, G. Papaioannou, E. Kalogeros, and R. Carter, "Representing Archeological Excavations Using the CIDOC CRM Based Conceptual Models," in Metadata and Semantic Research - 14th International Conference, MTSR 2020, Madrid, Spain, December 2-4, 2020, vol. 1355, pp. 355–366
- M. Gergatsoulis, G. Papaioannou, E. Kalogeros, I. Mpismpikopoulos, K. Tsiouprou, and R. Carter, "Modelling Archaeological Buildings Using CIDOC-CRM and Its Extensions: The Case of Fuwairit, Qatar", in Towards Open and Trustworthy Digital Societies - 23rd International Conference on Asia-Pacific Digital Libraries, ICADL 2021, Virtual Event, December 1–3, 2021, vol. 13133, pp. 357–372
- E. Kalogeros, M. Damigos, M. Sfakakis, S. Zapounidou, A. Drakopoulou, C. Zervopoulos, G. Martinis, C. Papatheodorou, and M. Gergatsoulis, "Digitizing, Transcribing and Publishing the Handwritten Music Score Archives of Ionian Islands Philharmonic Bands," in Metadata and Semantic Research - 15th International Conference, MTSR 2021, Virtual Event, November 29 - December 3, 2021, vol. 1537, pp. 370–381