Processing Aggregate Queries in a Federation of SPARQL Endpoints

DILSHOD IBRAGIMOV, KATJA HOSE, TORBEN BACH PEDERSEN, ESTEBAN ZIMÁNYI.

Motivating Example

- Earthquake in the Pacific in March 2011 ⟹ tsunami ⟹ a nuclear accident
- Hourly observation of radioactivity statistics at 47 prefectures
- Observations (March 16, 2011 – March 15, 2012) converted to RDF data (places represented by URI from GeoNames)
- Interesting analyses:
  - AVG radioactivity separately for each prefecture in Japan
  - The MIN and MAX radioactivity for each prefecture (changes within one-year observations)

Motivating Example - Observation

```
#observation
<http://www.kanzaki.com/works/2011/
    stat/ra/20110414/p13/t08>
  rdf:value "0.079"^^ms:microsv ;
  ev:place <http://sws.geonames.org/
    1852083/> ;
  ev:time <http://www.kanzaki.com/
    works/2011/stat/dim/d/
    20110414T08PT1H> ;
  scv:dataset <http://www.kanzaki.com/
    works/2011/stat/ra/set/moe> .
#dimension - place
<http://sws.geonames.org/1852083/>
  vcard:region "Tokyo"@en ;
  vcard:locality "Shinjuku"@en ;
  gn:lat "35.69355" ;
  gn:long "139.70352" .
#dimension - time
<http://www.kanzaki.com/works/2011/stat
    /dim/d/20110414T08PT1H>
  rdfs:label "2011-04-14T08";
  tl:at "2011-04-14T08:00:00+09:00"
    ^^xsd:dateTime ;
  tl:duration "PT1H"^^xsd:duration .
```

Motivating Example - Query

Ex: Show average radioactivity values for each prefecture

```
SELECT ?regName ( AVG (?floatRV ) AS ?average ) WHERE {
   ?s ev:place ?placeID .      ?s ev:time ?time .      ?s rdf:value ?radioValue .
   SERVICE <http://lod2.openlinksw.com/sparql> {
      ?placeID gn:parentFeature ?regionID .          ?regionID gn:name ?regName .
   }
   BIND (xsd:float (?radioValue ) as ?floatRV ) .
}
GROUP BY ?regName
```

Motivating Example - Results

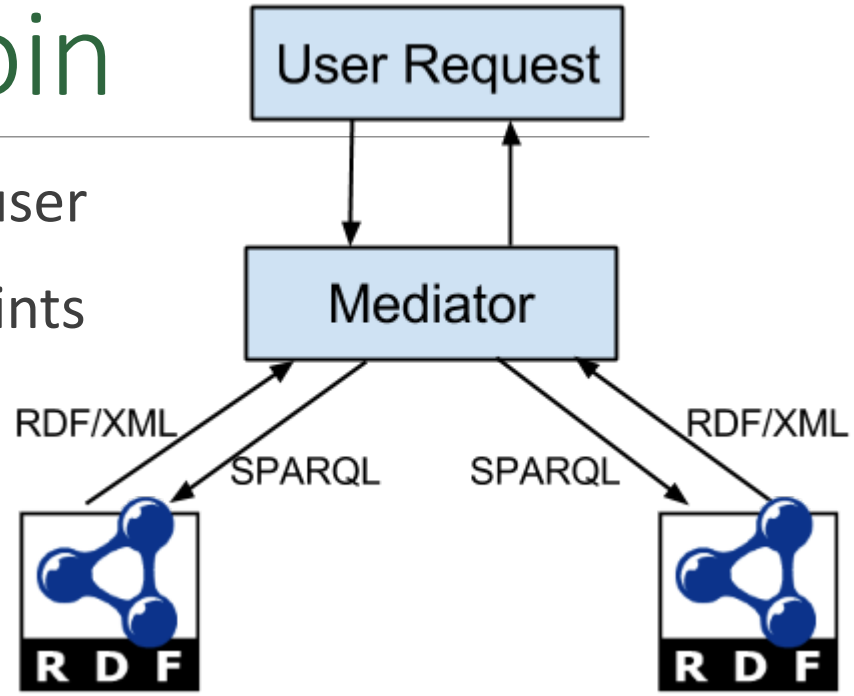
- Virtuoso v07.10.3207, Sesame v2.7.11, and Jena Fuseki v1.0.0 (based on ARQ) timed out
- Network traffic analyzer showed that:
  - Virtuoso and Fuseki query GeoNames for each radioactivity observation (more than 400,000 requests)
  - Sesame is trying to download all triples that match the SERVICE query pattern (more than 7.8 million triples)

Outline

- Basic Strategies
- CODA
- Test Case
- Conclusion and Future Works

Basic Strategies - Mediator Join

- The mediator/federator receives the query from the user
- The query optimizer sends separate queries to endpoints and merges the results
- Strong point – parallelization
- Weak point – expensive for large intermediate results/datasets




```
SELECT ?placeID ?radioValue WHERE {
  ?s ev:place ?placeID; ev:time ?time.
  ?s rdf:value ?radioValue.
}
```

```
SELECT ?placeID ?regName WHERE {
  ?placeID gn:parentFeature ?regionID.
  ?regionID gn:name ?regName.
}
```

Basic Strategies - Semi-Join

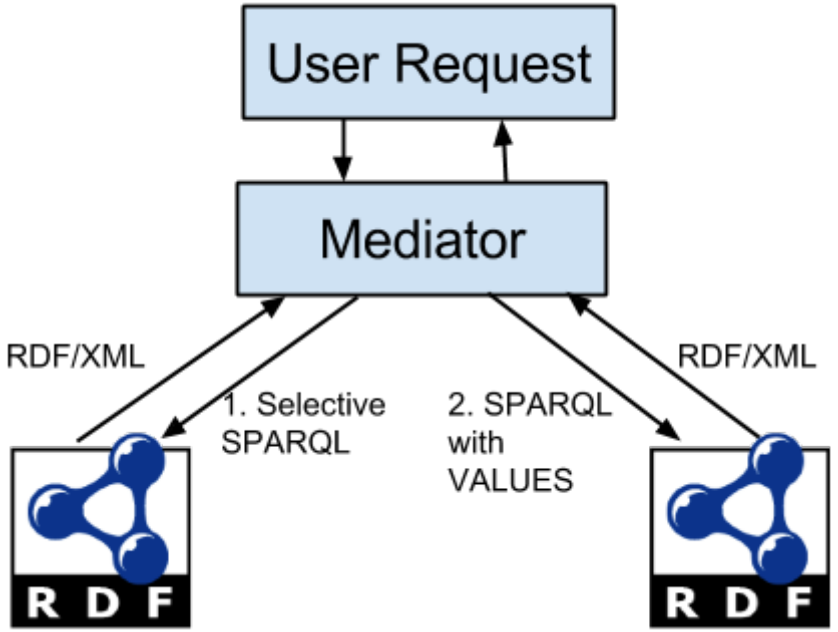
- Main principle is to execute the subquery with the smallest result first and use the retrieved results as bindings for the join variable in other subqueries (SPARQL structure)
- Efficient for highly selective subqueries (with FILTER statement)

```
SELECT ?regName ( AVG (?radioValue ) AS ?average ) WHERE {
    ?s ev:place ?placeID .      ?s ev:time ?time .      ?s rdf:value ?radioValue .
    SERVICE <http://lod2.openlinksw.com/sparql>{
        ?placeID gn: parentFeature ?regionID .      ?regionID gn:name ?regName .
    }
    FILTER(?radioValue < 0.08) .
} GROUP BY ?regName
```

Basic Strategies - Semi-Join (Cont)

```
SELECT ?placeID ?radioVal
WHERE {
  ?s rdf:value ?radioVal ;
  ev:place ?placeID; ev:time ?time.
  FILTER (?radioValue < 0.08) .
}
```

```
SELECT ?placeID ?regName
WHERE { ?placeID gn:parentFeature ?rgID.
  ?rgID gn:name ?regName.
  VALUES (?placeID) {
  <http://sws.geonames.org/1852083/>...}
}
```



- Weak point - VALUES is not yet widely adopted in existing endpoints. SPARQL 1.0 compliant alternatives of UNION (or FILTER) must often be used

Basic Strategies - Partial Aggregation

- If results are grouped by SERVICE query variables, further optimization is possible (motivating query example)

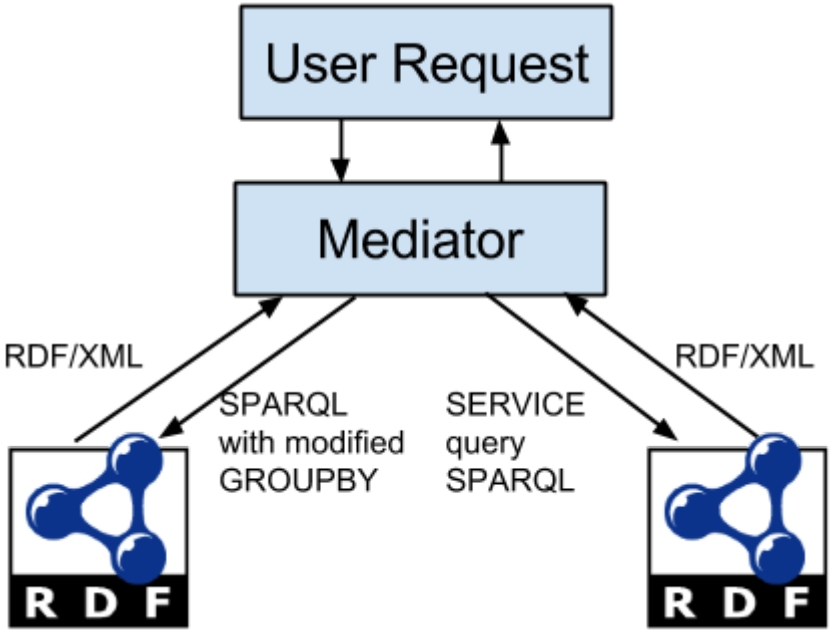
1) First group by the observation place (?placeID)

```
SELECT ?placeID (SUM (?floatRV ) AS ?avgSUM) (COUNT (?floatRV ) AS ?avgCNT )
WHERE {
   ?s ev:place ?placeID .     ?s ev:time ? time .     ?s rdf:value ? radioValue .
   BIND (xsd:float (?radioValue ) as ?floatRV ) .
}
GROUP BY ?placeID
```

Basic Strategies - Partial Aggregation (Cont)

- *Then execute SERVICE query*

```
SELECT ?placeID ?regName WHERE {
    ?placeID gn:parentFeature ?regionID .
    ?regionID gn:name ?regName .
    VALUES (?placeID) {
        (<http://sws.geonames.org/1852083/>) ….
    }
}
```



- *Final step – join the intermediate results and compute the final result* (distributed/algebraic functions)

CODA – Cost-based Optimizer for Distributed Aggregate Queries

- Decomposes the original query into multiple subqueries (query Q_M and SERVICE queries Q_{e1} … Q_{eN})

- Estimates query execution costs for different query execution plans

- Chooses the one with minimum costs

CODA - Costs

- Overall costs C_Q

$$C_Q = C_P + C_C$$

- Communication costs C_C for subquery S_i:

$C_C(S_i) = C_O + c_{S_i} * C_{map}$; C_O - communication establishing overhead , c_{S_i} - result size, and C_{map} - single result transfer cost

- Processing costs

$C_P = c_{agg_i} * C_{AGG}$; c_{agg_i} - number of aggregated observations, C_{AGG} - cost for processing a single observation

CODA - Estimating Constants

- C_{map} - estimated using “SELECT * WHERE { ?s #p ?o . FILTER(?o = #o) } LIMIT #L”; different values for #L, #o and #p
- C_O - estimated with multiple “ASK {}” or “SELECT (1 AS ?v) {}”
- C_{AGG} - estimated based on multiple “SELECT COUNT(?s) WHERE {?s ?p ?o } GROUP BY ?o”

Not perfectly accurate but the aim is to find out which execution plan is more efficient (not to predict the execution costs)

CODA - Result Size Estimation

- Result size estimation - VoID statistics (dataset, property partition, class partition)
- c_t - total number of triples (void:triples), c_s- total number of distinct subjects (void:distinctSubjects), c_o - total number of distinct objects (void:distinctObjects)
- Single patterns - C_{res} for (?s ?p ?o) is given by c_t, (s ?p ?o) estimated as c_t/c_s, (?s ?p o) as c_t/c_o, and (s ?p o) as $c_t/(c_s * c_o)$; FILTER influence estimates
- Joins - estimates depend on shape (star vs path). Formulas taken from "Resource Planning for SPARQL Query Execution on Data Sharing Platforms".

type	**pattern**	**cardinality** $card(join, partition)$
subject – subject	?s predA ?o . ?s predB ?o2	$\frac{card(pat_1) \cdot card(pat_2)}{\max(cp_{predA,s}, cp_{predB,s})}$

CODA – Motivating Example

- Decomposed into 3 queries

```
SELECT ?placeID (AVG(?floatRV) AS ?average)
WHERE
{
  ?s ev:place ?placeID .
  ?s rdf:value ?radioValue .
  BIND(xsd:float(?radioValue) AS ?floatRV) .
  ?s ev:time ?time .
}
GROUP BY ?placeID
```

```
SELECT ?placeID ?floatRV
WHERE
{
  ?s ev:place ?placeID .
  ?s rdf:value ?radioValue .
  BIND(xsd:float(?radioValue) AS ?floatRV) .
  ?s ev:time ?time .
}
```

```
SELECT ?placeID ?regName
WHERE
{
  ?placeID gn:parentFeature ?regionID .
  ?regionID gn:name ?regName .
}
```

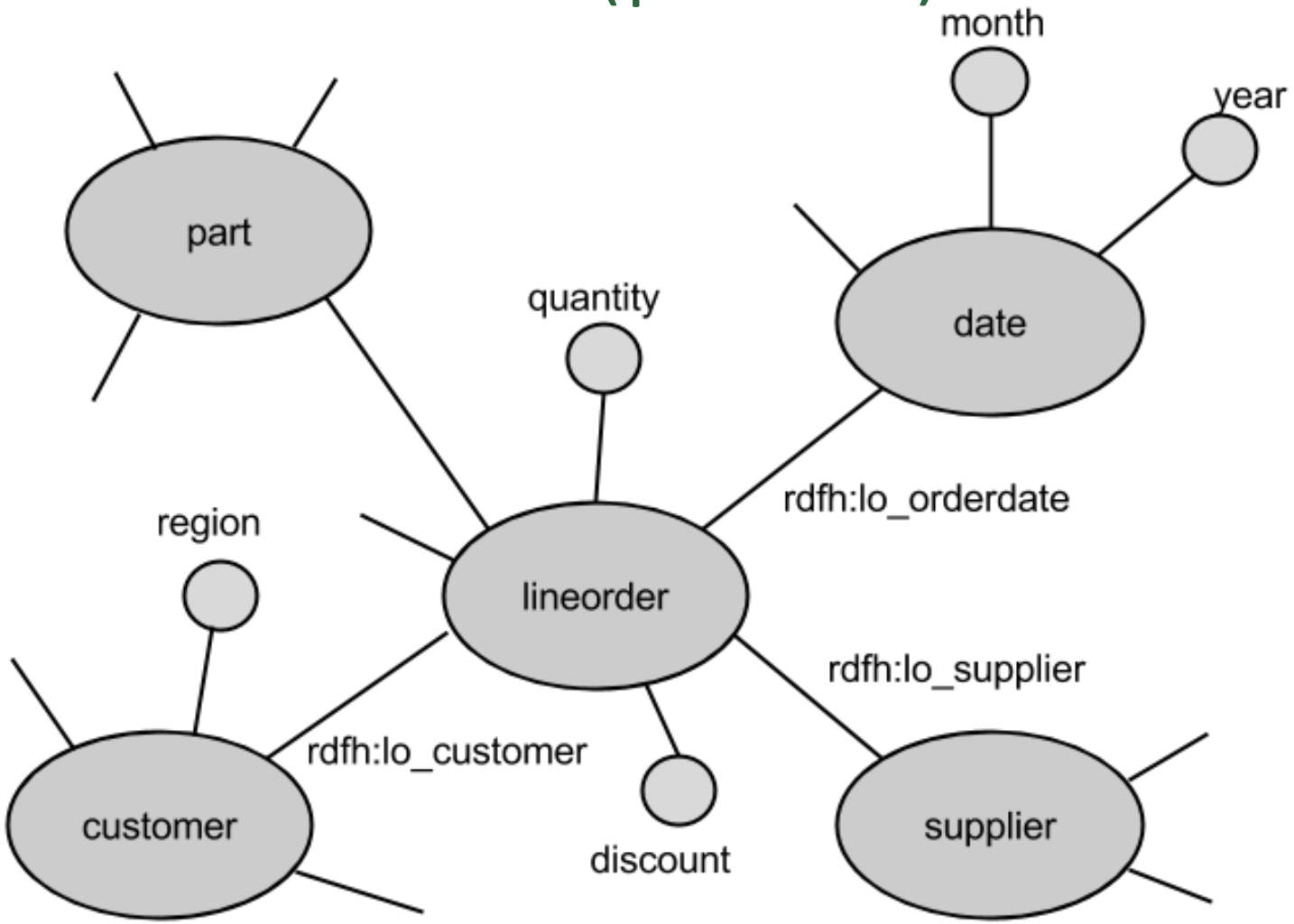
CODA – Motivating Example

- Estimates for Radioact query:
  - number of aggregated triples: 405384
  - estimated cost: 15
  - number of returned triples: 405384
  - estimated cost: 129
- Estimates for GeoNames query:
  - number of returned triples: 7877627
  - estimated cost: 1956
- Selected plan – Partial Aggregation

Test Case – SSB as RDF

- Star Schema Benchmark converted to RDF (strongly resembling SSB tabular structure)
- We generated data for different scale factors (1 to 5 - 6M to 30M observations, 110,5M to 547,5M triples)
- Different configurations
  - two endpoints (one endpoint containing main observation data and one SERVICE endpoint containing supporting data)
  - three endpoints (two SERVICE endpoints containing supporting data)
  - four endpoints (three SERVICE endpoints containing supporting data)
- All datasets and queries are available at http://extbi.cs.aau.dk/coda/

SSB RDF schema (partial)



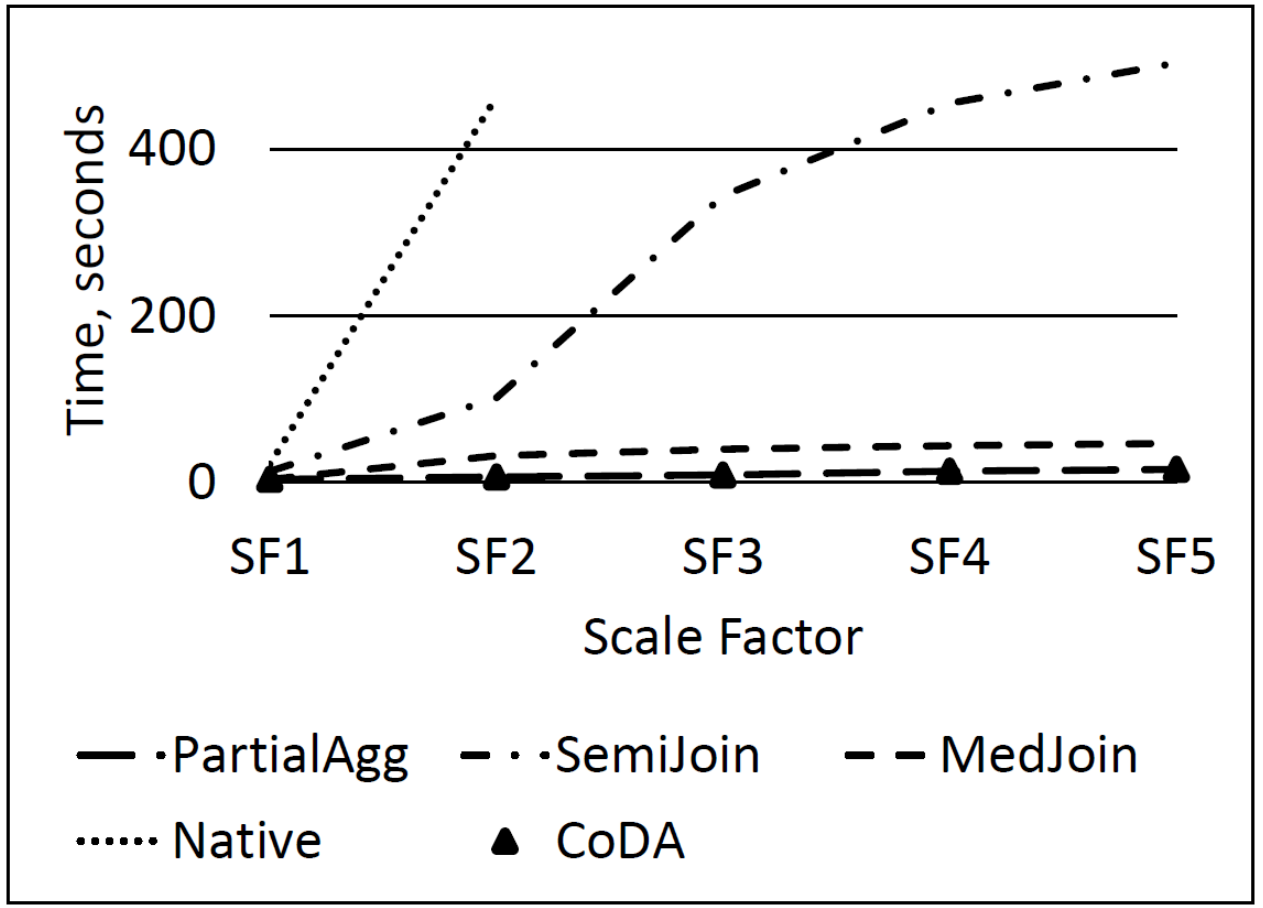
Test Case – SSB Queries

Query Prototypes	Query No	Query Parameters for Various Selectivities
Prototype 1. Amount of revenue increase that would have resulted from eliminating certain company-wide discounts.	Q1.1	Discounts 1, 2, and 3 for quantities less than 25 shipped in 1993.
	Q1.2	Discounts 1, 2, and 3 for quantities less than 25 shipped in 01/1993.
	Q1.3	Discounts 5, 6, and 7 for quantities less than 35 shipped in week 6 of 1993.
Prototype 2. Revenue for some product classes, for suppliers in a certain region, grouped by more restrictive product classes and all years.	Q2.1	Revenue for 'MFGR#12' category, for suppliers in America
	Q2.2	Revenue for brands 'MFGR#2221' to 'MFGR#2228', for suppliers in Asia
	Q2.3	Revenue for brand 'MFGR#2239' for suppliers in Europe
Prototype 3. Revenue for some product classes, for suppliers in a certain region, grouped by more restrictive product classes and all years.	Q3.1	For Asian suppliers and customers in 1992-1997
	Q3.2	For US suppliers and customers in 1992-1997
	Q3.3	For specific UK cities suppliers and customers in 1992-1997
	Q3.4	For specific UK cities suppliers and customers in 12/1997
Prototype 4. Aggregate profit, measured by subtracting revenue from supply cost.	Q4.1	For American suppliers and customers for manufacturers 'MFGR#1' or 'MFGR#2' in 1992
	Q4.2	For American suppliers and customers for manufacturers 'MFGR#1' or 'MFGR#2' in 1997-1998
	Q4.3	For American customers and US suppliers for category 'MFGR#14' in 1997-1998

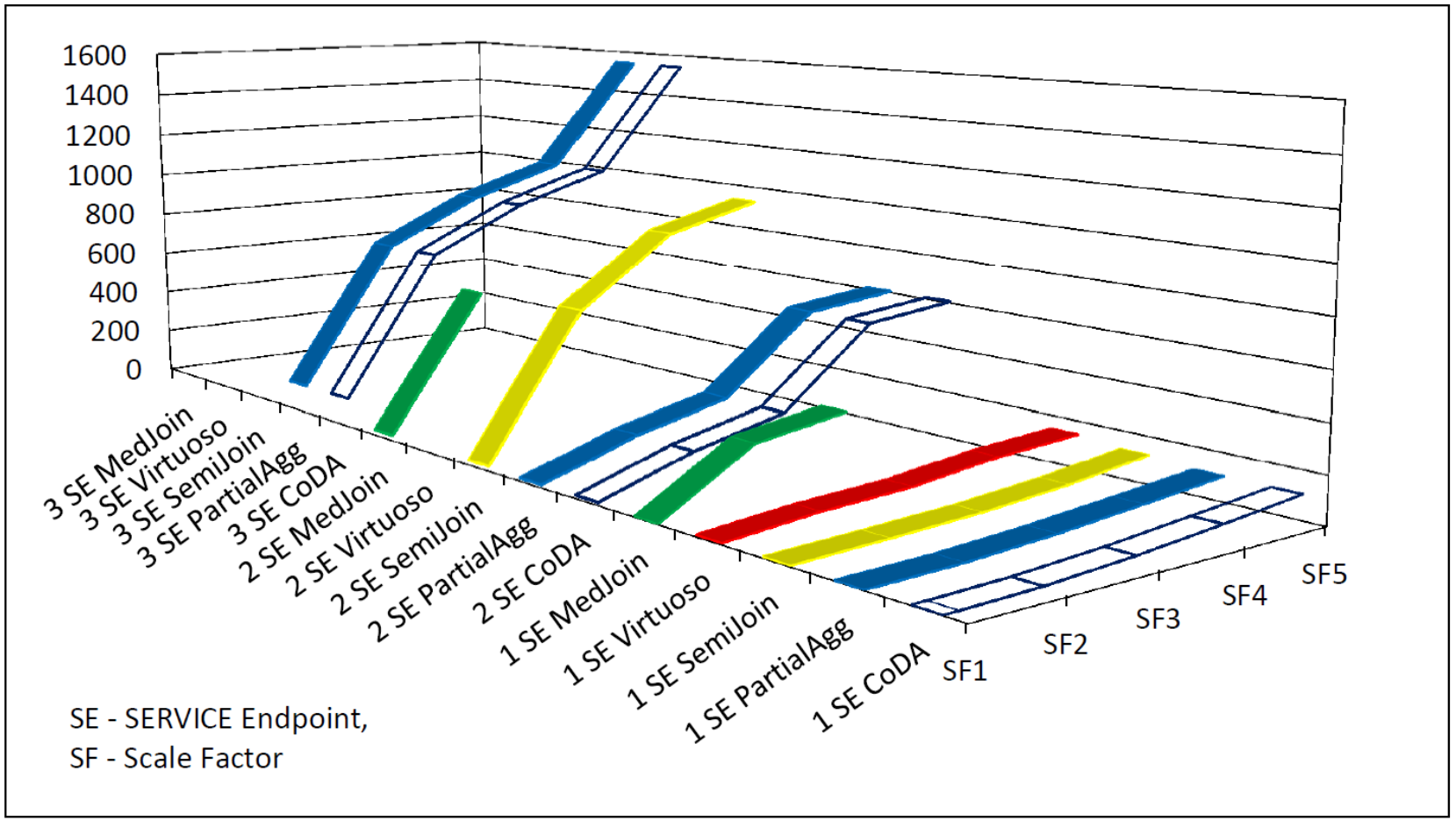
Test Case – Results (One SERVICE Endpoint)

	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3
Scale Factor 1													
Virtuoso	T/O	T/O	760	500,3	107,8	21,3	215,8	21,2	1,4	1,4	863	969	7,3
SemiJoin	**1,3**	**0,2**	**0,1**	12,7	13,5	12,6	**14,1**	11,0	6,5	**0,2**	4,8	**8,1**	4,5
PartialAgg	1,6	1,4	0,8	**9,4**	**4,5**	3	17,5	**2,8**	**0,5**	0,3	**4**	18,5	**1,0**
MedJoin	249,5	213,4	82,9	11	5,2	**2,9**	98,9	3,4	0,8	0,3	26,4	32	1,1
CoDA	**1,3**	**0,2**	**0,1**	**9,4**	**4,5**	**2,9**	**14,1**	**2,8**	**0,5**	**0,2**	**4**	**8,1**	**1,0**

Test Case – Results (One SERVICE Endpoint Q2.3)



Test Case – Results (One to Three SERVICE Endpoints Q4.3)



Conclusion and Future Work

- Efficiently processing aggregate queries in a federation of SPARQL endpoints
- Processing strategies (MedJoin, SemiJoin, PartialAgg)
- Cost-based Optimizer for Distributed Aggregate queries (CODA)
  - efficient and scalable
  - chooses the best query processing plan in different situations
  - significantly outperforms current state-of-the art triple stores
- Future Work:
  - Using more complex statistics with precomputed join result sizes and correlation information for better cardinality estimation
  - Optimizing more complex queries, e.g., with optional patterns or complex aggregation functions