A FINE-GRAINED EVALUATION OF SPARQL ENDPOINT FEDERATION SYSTEMS

Muhammad Saleem, Yasar Khan, Ali Hasnain, Ivan Ermilov, Axel-Cyrille Ngonga Ngomo

ISWC 2016, Kobe, Japan, 20/10/2016

Agile Knowledge Engineering and Semantic Web (AKSW), University of Leipzig, Germany

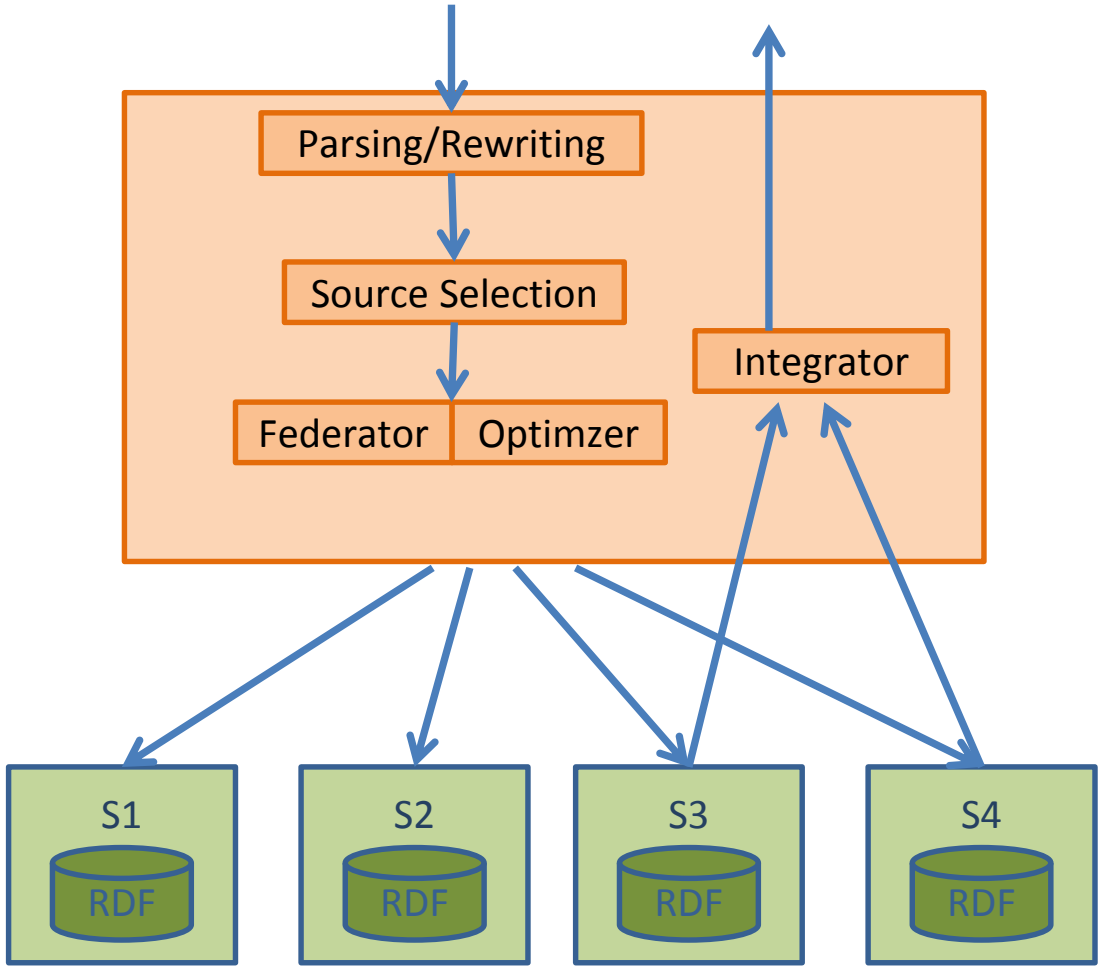
AGENDA

- SPARQL query federation
- Survey of federated SPARQL query processing systems
- Performance variables
- Evaluation of SPARQL endpoint federation systems

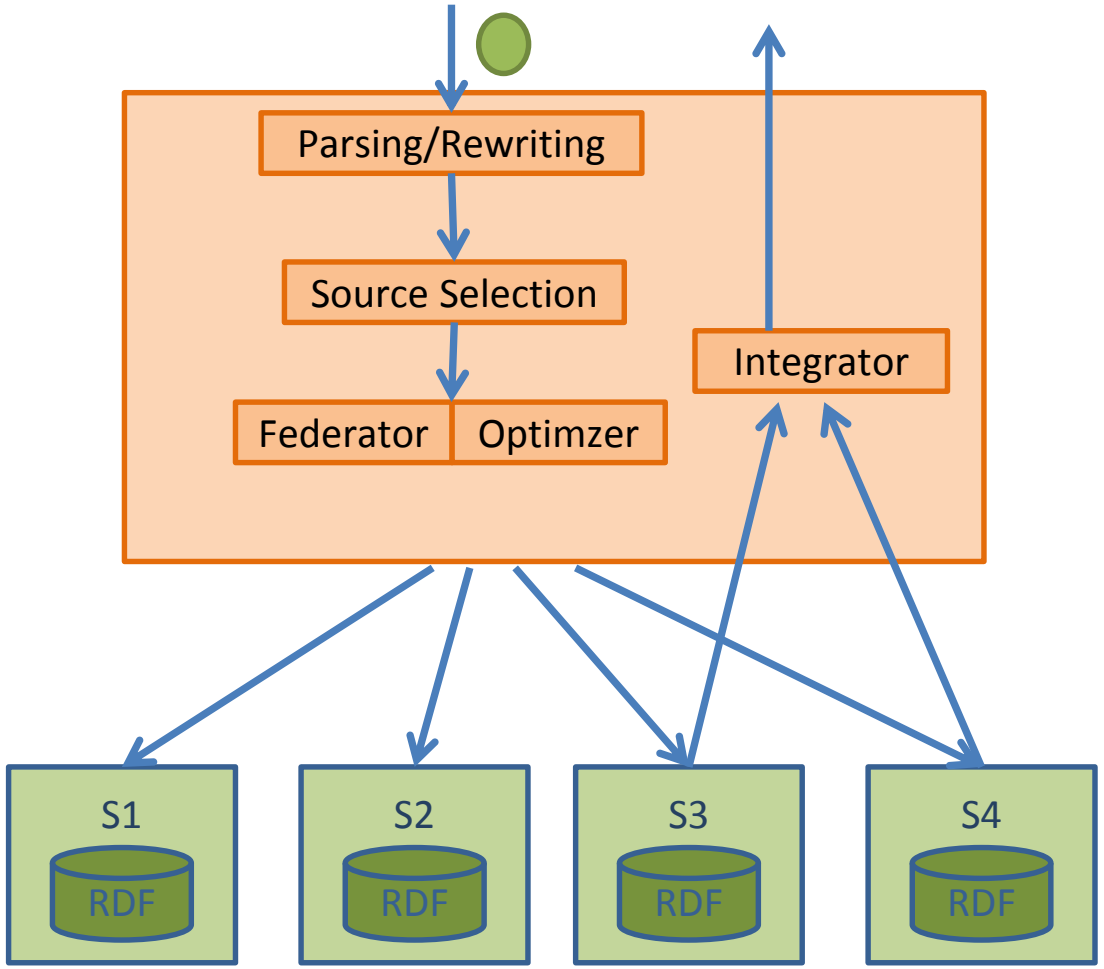
SPARQL QUERY FEDERATION APPROACHES

- SPARQL Endpoint Federation (SEF)
- Linked Data Federation (LDF)
- Hybrid of SEF+LDF

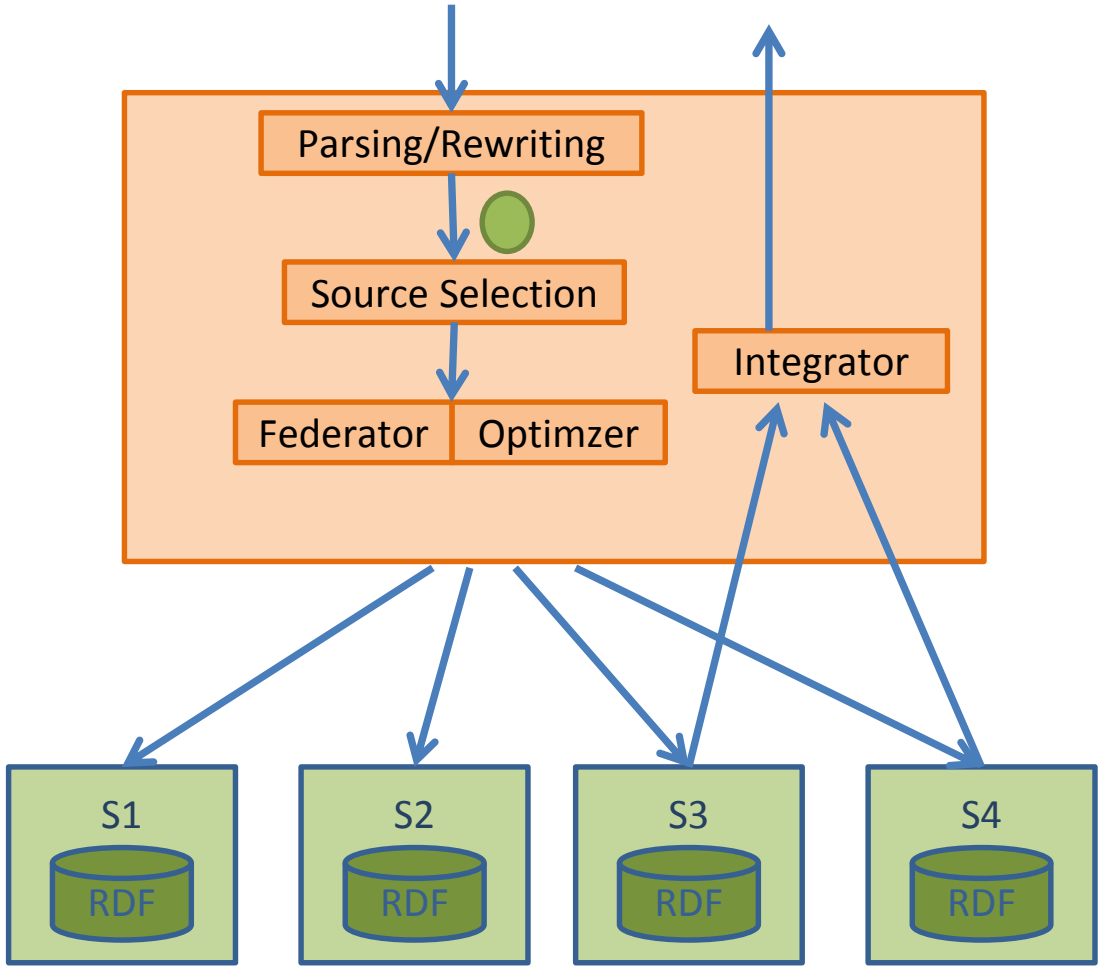
SPARQL Endpoint Federation



SPARQL Endpoint Federation

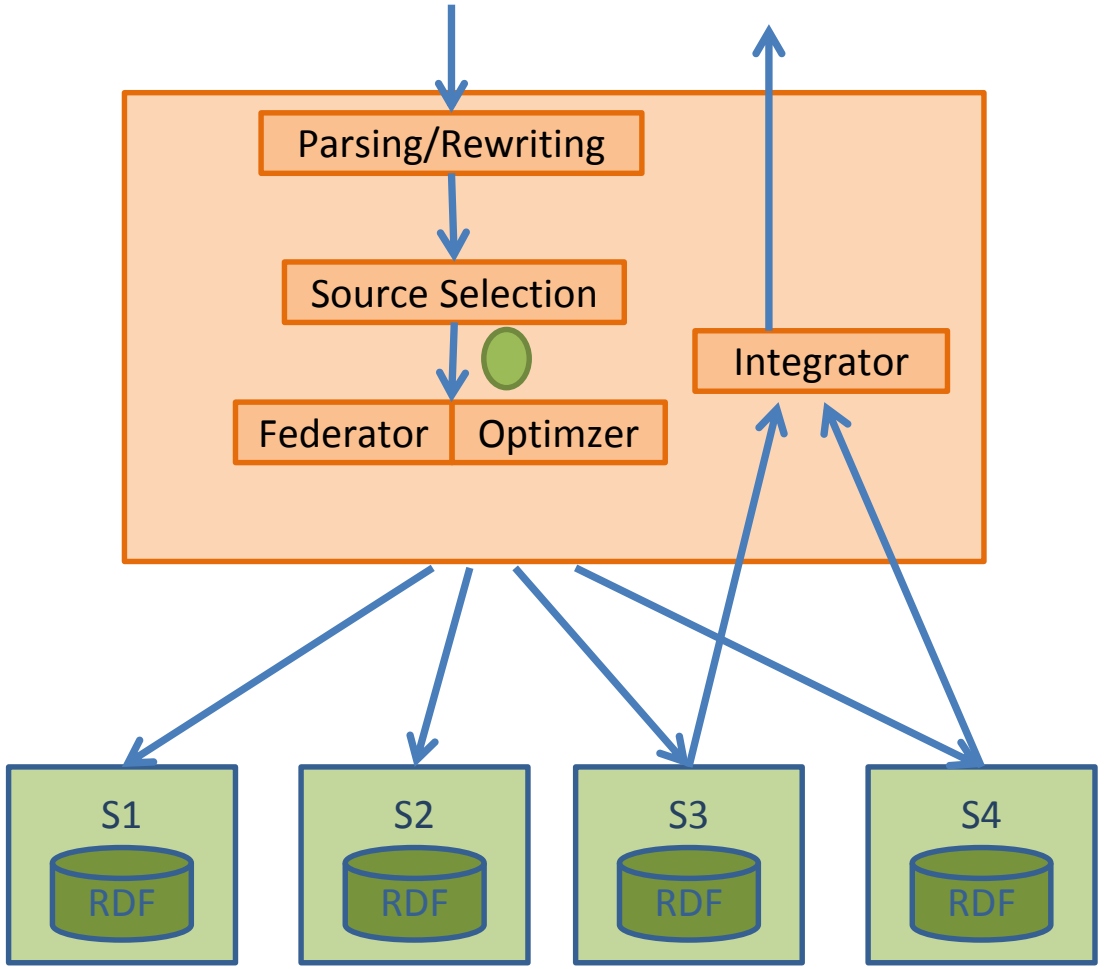


SPARQL Endpoint Federation



Rewrite query and get Individual Triple Patterns

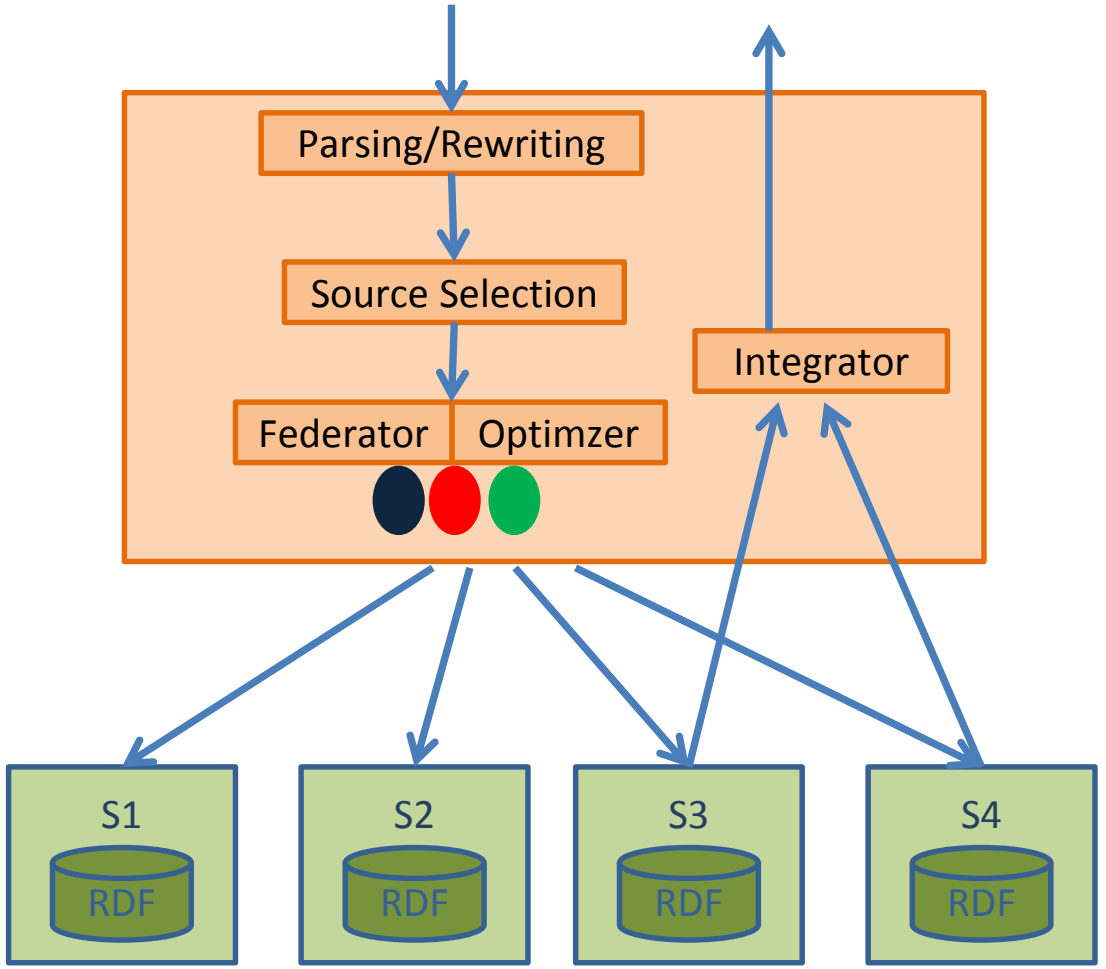
SPARQL Endpoint Federation



Rewrite query and get Individual Triple Patterns

Identify capable source against Individual Triple Patterns

SPARQL Endpoint Federation

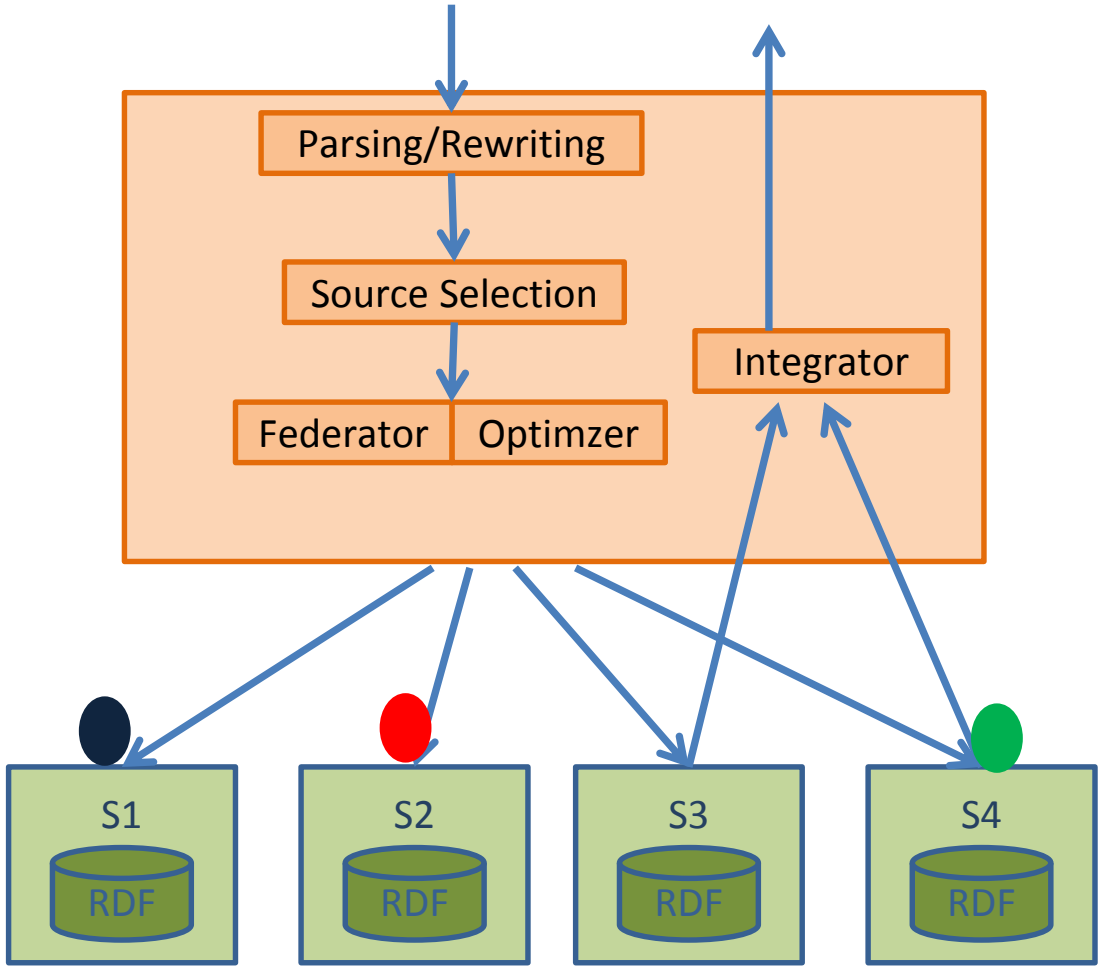


Rewrite query and get Individual Triple Patterns

Identify capable source against Individual Triple Patterns

Generate optimized sub-query Exe. Plan

SPARQL Endpoint Federation



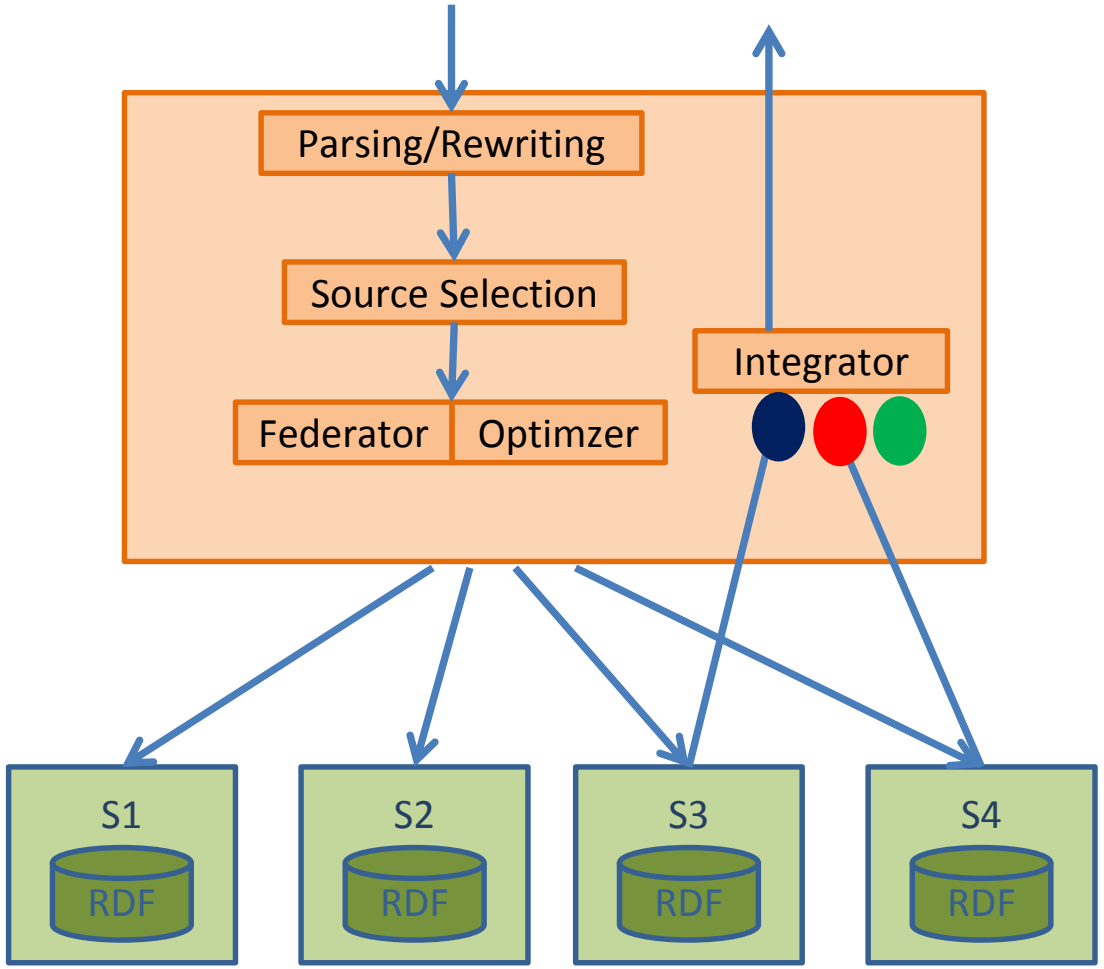
Rewrite query and get Individual Triple Patterns

Identify capable source against Individual Triple Patterns

Generate optimized sub-query Exe. Plan

Execute sub-queries

SPARQL Endpoint Federation



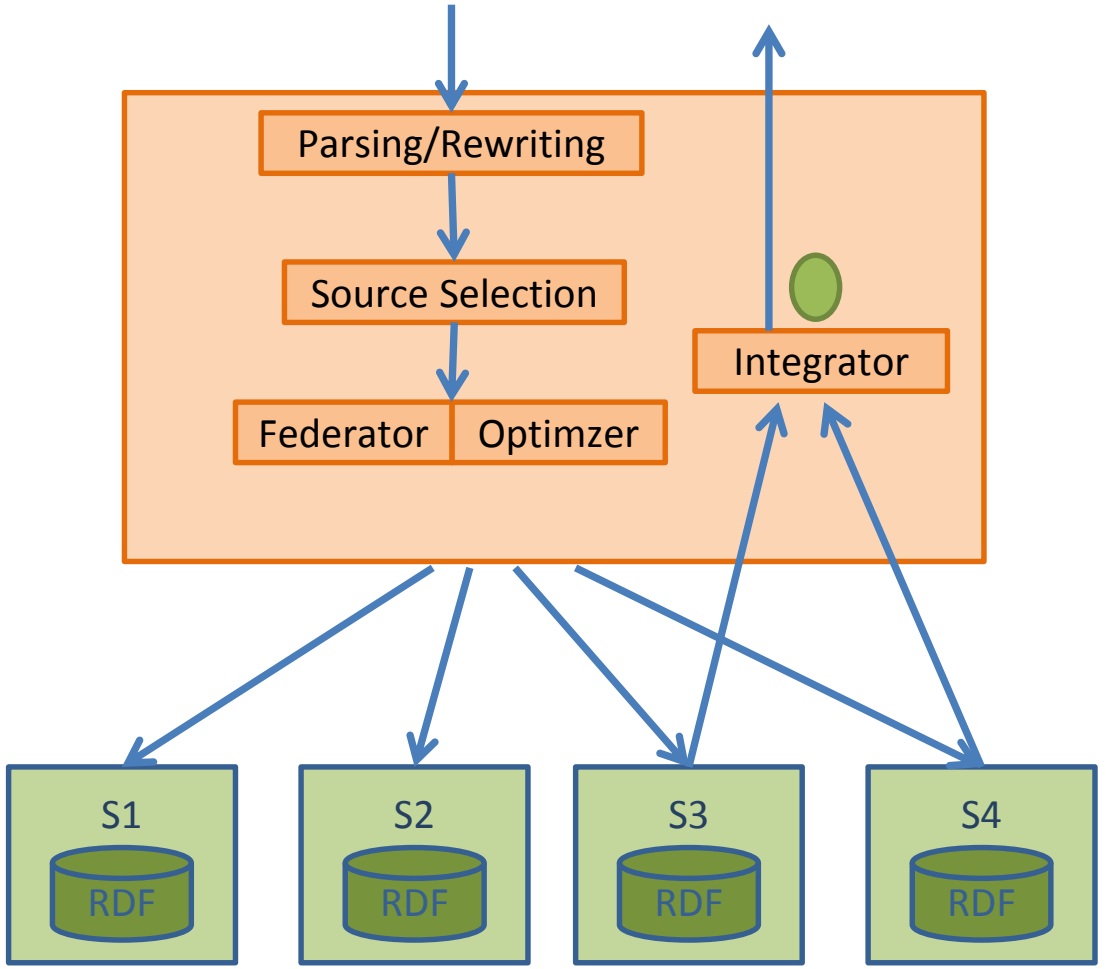
Rewrite query and get Individual Triple Patterns

Identify capable source against Individual Triple Patterns

Generate optimized sub-query Exe. Plan

Execute sub-queries

SPARQL Endpoint Federation



Rewrite query and get Individual Triple Patterns

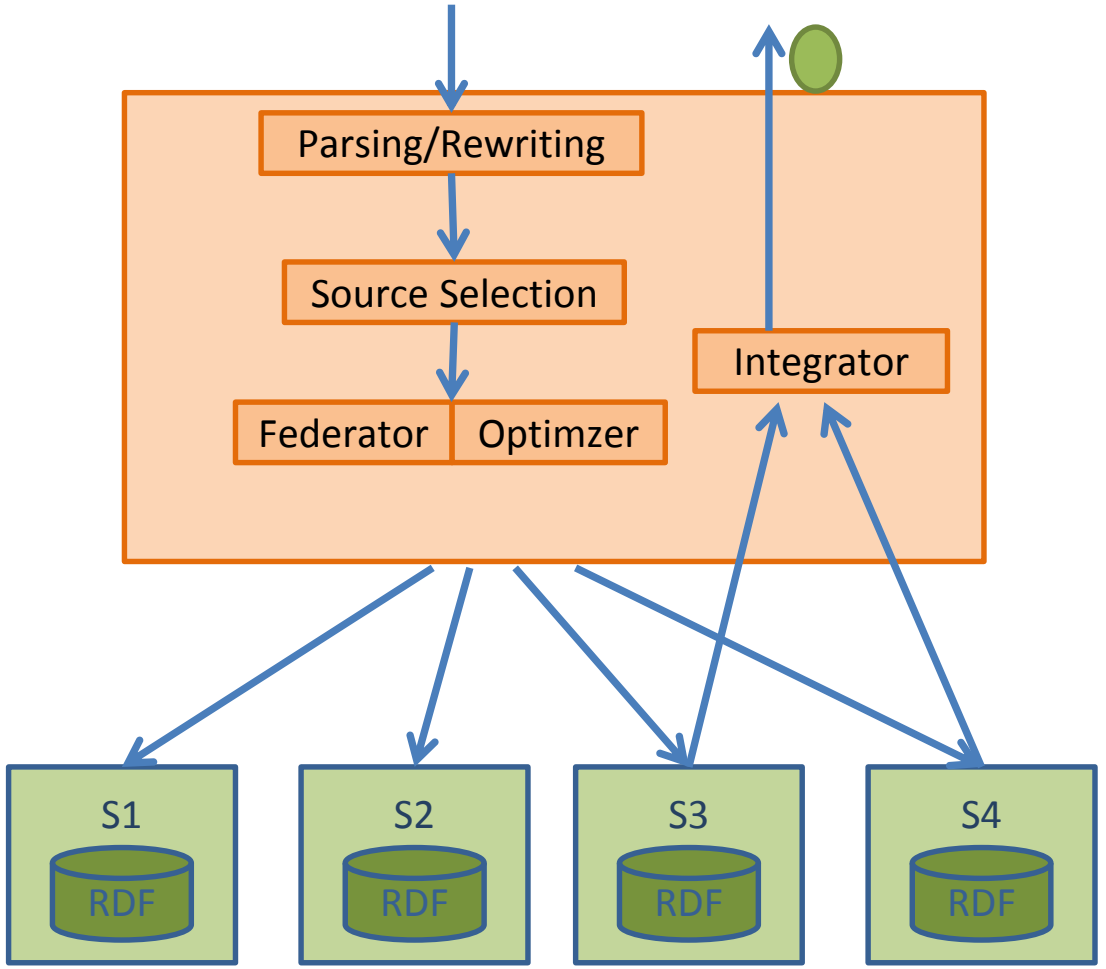
Identify capable source against Individual Triple Patterns

Generate optimized sub-query Exe. Plan

Execute sub-queries

Integrate sub-queries results

SPARQL Endpoint Federation



Rewrite query and get Individual Triple Patterns

Identify capable source against Individual Triple Patterns

Generate optimized sub-query Exe. Plan

Execute sub-queries

Integrate sub-queries results

TYPES OF SOURCE SELECTION

- Index-free
  - Using SPARQL ASK queries
  - Potentially ensures result set completeness
  - SPARQL ASK queries can be expensive
  - SPARQL ASK cache is beneficial
  - E.g., FedX
- Index-only
  - Only make use of Index/data summaries
  - Less efficient but fast source selection
  - Result set completeness is not ensured
  - E.g., DARQ, LHD

TYPES OF SOURCE SELECTION

- Hybrid
  - Make use of index+SPARQL ASK
  - Most efficient
  - Result set completeness is not ensured
  - SPARQL ASK cache is beneficial
  - E.g., HiBISCuS, ANAPSID, SPLENDID

SURVEY: FEDERATED SPARQL ENGINES

- System Information
  - Title and/or URL of federation engine
  - Code available?
  - Implementation and license
  - Type of source selection
  - Type of join(s) used for data integration
  - Use of cache?
  - Support for catalog/index update

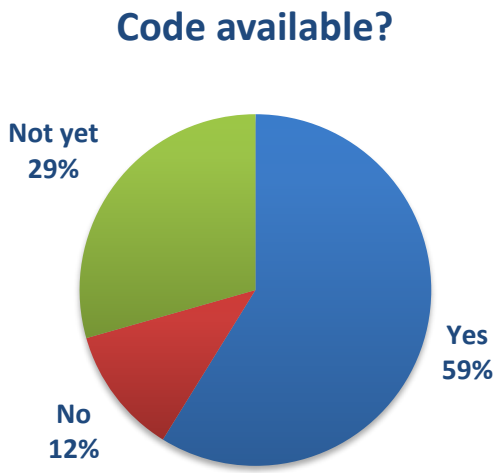
SURVEY: FEDERATED SPARQL ENGINES

C.A. = Code Availability, **S.S.T.** = Source Selection Type, **I.U.** = Index/catalog Update, **AGJ** = Adaptive Group Join, **ADJ** = Adaptive Dependent Join

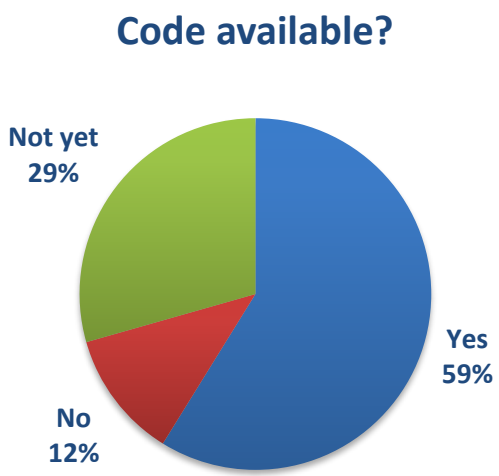
Systems	Category	C.A	Implementation	Licencing	S.S.T	Join Type	Cache	I.U
FedX	SEF	✓	Java	GNU A.G.P.L	index-free	bind (VENL)	✓	NA
LHD	SEF	✓	Java	MIT	hybrid (A+I)	hash/ bind	✗	✗
SPLENDID	SEF	✓	Java	L.G.P.L	hybrid (A+I)	hash/ bind	✗	✗
FedSearch	SEF	✗	Java	GNU A.G.P.L	hybrid (A+I)	bind , pull based rank join (RMHJ)	✓	NA
GRANATUM	SEF	✗	Java	yet to decide	index only	nested loop	✗	✗
Avalanche	SEF	✗	Python, C, C++	yet to decide	index only	distributed, merge	✓	✗
ANAPSID	SEF	✓	Python	GNU G.P.L	hybrid (A+I)	AGJ, ADJ	✗	✓
ADERIS	SEF	✓	Java	Apache	Index only	index-based nested loop	✗	✗
DARQ	SEF	✓	Java	GPL	Index only	nested loop, bound	✗	✗
LDQPS	LDF	✗	Java	Scala	hybrid (C+L)	symmetric hash	✗	✗
SIHJoin	LDF	✗	Java	Scala	hybrid (C+L)	symmetric hash	✗	✗
WoDQA	LDF	✓	Java	GPL	hybrid (A+I)	nested loop, bound	✓	✓
Atlas	DHTF	✓	Java	GNU L.G.P.L	Index only	SQLite	✗	✗
DAW	SEF	✗	Java	GNU G.P.L	hybrid (A+I)	based on underlying system	✓	✗
SAFE	SEF	✓	Java	GNU G.P.L	hybrid (A+I)	bind (VENL)	✓	✓
QUETSAL	SEF	✗	Java	GNU G.P.L	hybrid (A+I)	based on Sesame	✓	✓
HiBISCuS	SEF	✓	Java	GNU G.P.L	hybrid (A+I)	based on underlying system	✓	✓

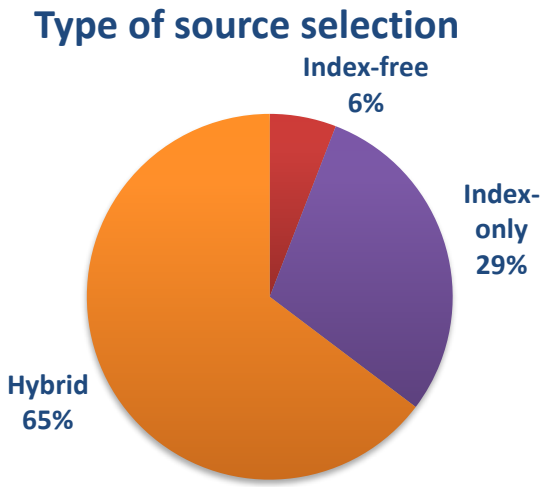
SURVEY: FEDERATED SPARQL ENGINES

SURVEY: FEDERATED SPARQL ENGINES

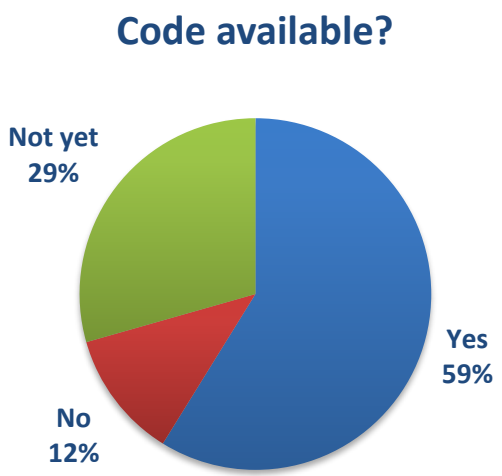


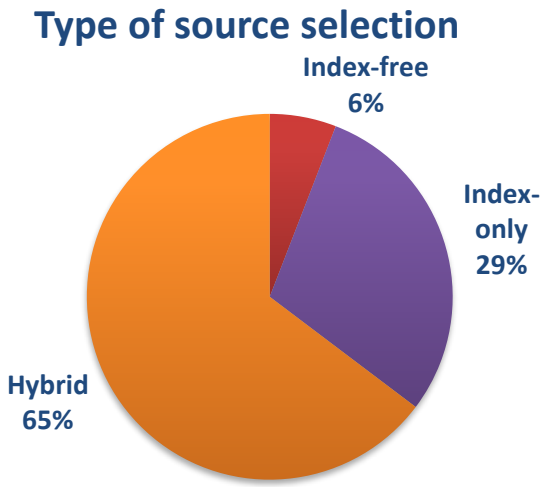
SURVEY: FEDERATED SPARQL ENGINES

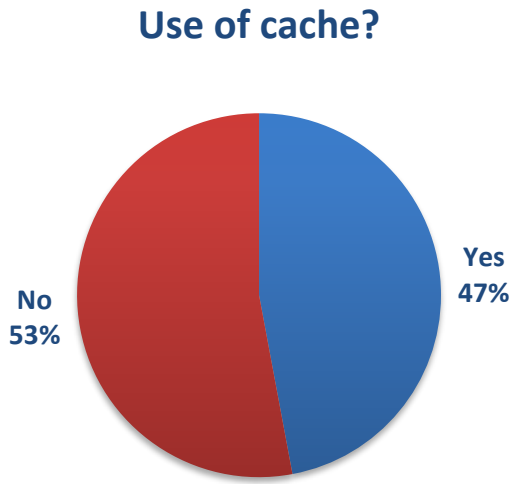




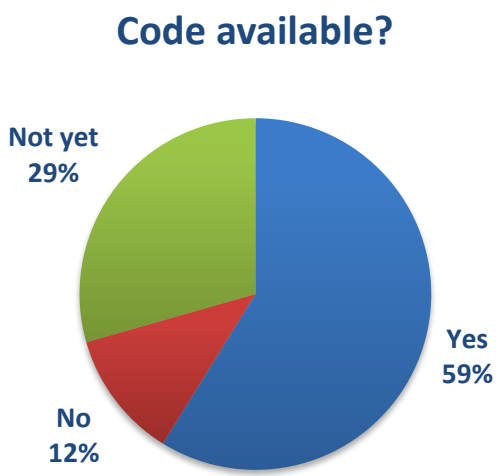
SURVEY: FEDERATED SPARQL ENGINES

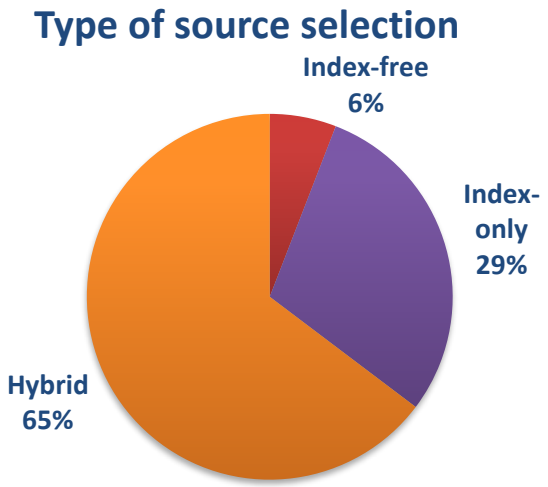


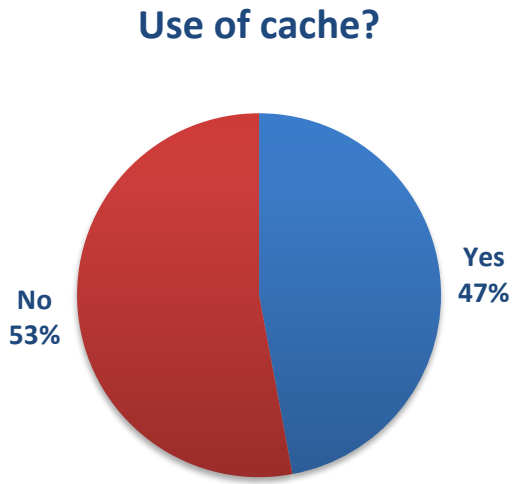


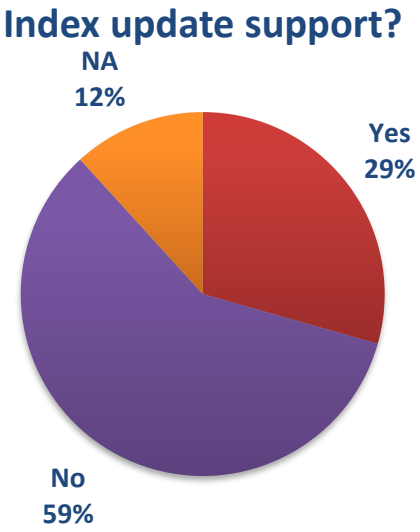


SURVEY: FEDERATED SPARQL ENGINES









SURVEY: FEDERATED SPARQL ENGINES

- Requirements
  - Result completeness
  - Policy-based query planning
  - Support for partial results retrieval
  - Support for no-blocking operator/ adaptive query processing
  - Support for provenance information
  - Query runtime estimation
  - Duplicate Detection
  - Top-K query processing
  - Supported SPARQL types/ clauses

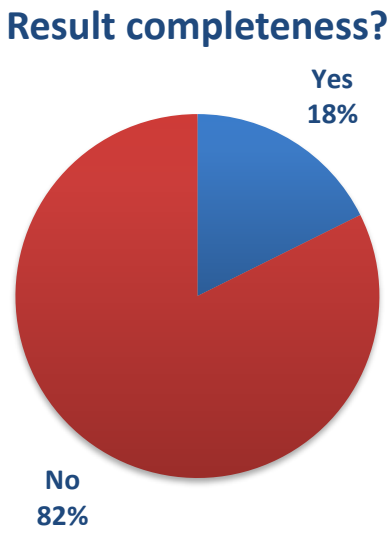
SURVEY: FEDERATED SPARQL ENGINES

R.C. = Results Completeness, **P.R.R.** = Partial Results Retrieval, **N.B.O.** = No Blocking Operator, **A.Q.P.** = Adaptive Query Processing, **D.D.** = Duplicate Detection, **P.B.Q.P** = Policy-based Query Planning, **Q.R.E.** = Query Runtime Estimation, **Top-K.Q.P** = Top-K query processing

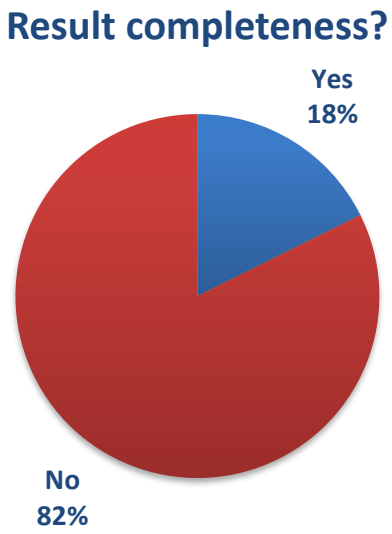
Systems	R.C.	P.R.R.	N.B.O / A.Q.P.	D. D.	P.B.Q.P	Provenance	Q.R.E	Top-K.Q.P
FedX	✓	✗	✗	✗	✗	✗	✗	✗
LHD	✗	✗	✗	✗	✗	✗	✗	✗
SPLENDID	✗	✗	✗	✗	✗	✗	✗	✗
FedSearch	✓	✗	✗	✗	✗	✗	✗	✗
GRANATUM	✗	✗	✗	✗	partial	partial	✗	✗
Avalanche	✗	✓	✓	partial	✗	✗	✗	✗
ANAPSID	✗	✗	✓	✗	✗	✗	✗	✗
ADERIS	✗	✗	✓	✗	✗	✗	✗	✗
DARQ	✗	✗	✗	✗	✗	✗	✗	✗
LDQPS	✗	✗	✓	✗	✗	✗	✗	✗
SIHJoin	✗	✗	✓	✗	✗	✗	✗	✗
WoDQA	✓	✗	✗	✗	✗	✗	✗	✗
Atlas	✗	✗	✗	partial	✗	✗	✗	✗
DAW	✗	✓	based on underlying system	✓	✗	✗	✗	✗
SAFE	✗	✗	✗	✗	✓	✗	✗	✗
QUETSAL	✗	✓	✗	✓	✗	✗	✗	✓
HiBISCuS	✗	✓	based on underlying system	✗	✗	✗	✗	✗

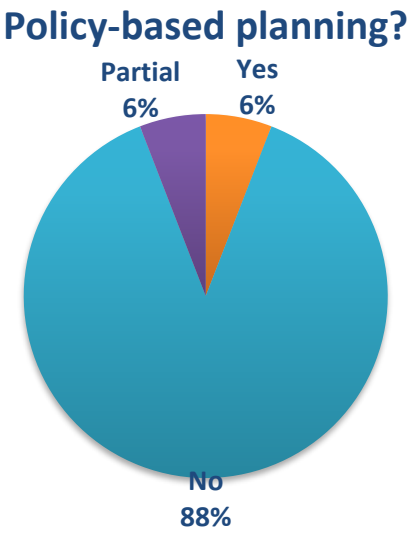
SURVEY: FEDERATED SPARQL ENGINES

SURVEY: FEDERATED SPARQL ENGINES

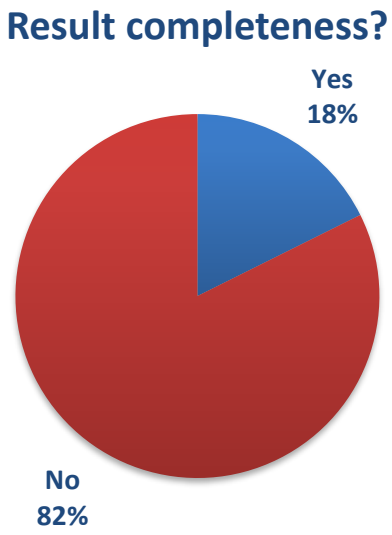


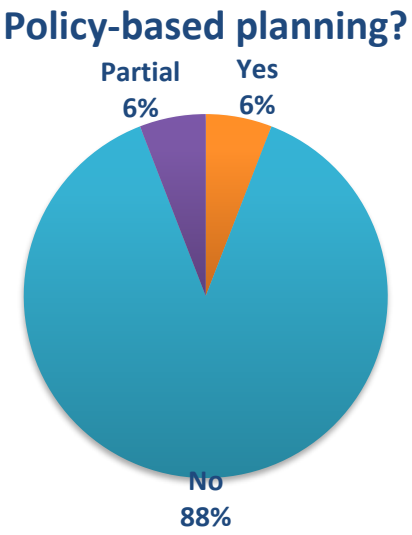
SURVEY: FEDERATED SPARQL ENGINES

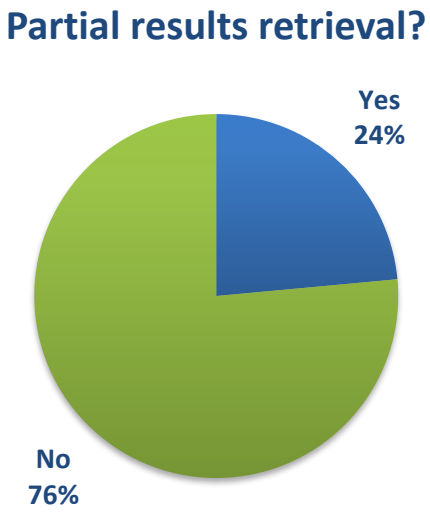




SURVEY: FEDERATED SPARQL ENGINES

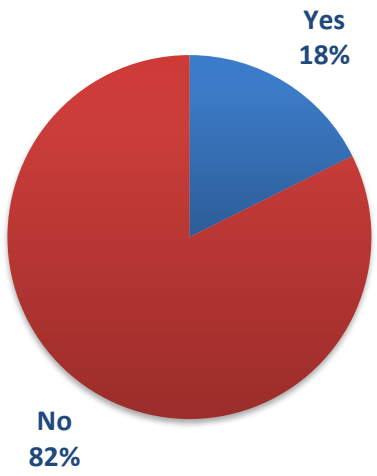




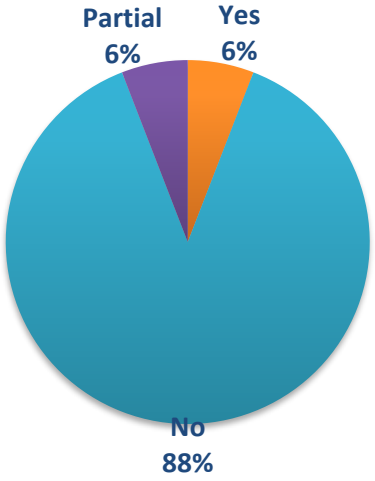


SURVEY: FEDERATED SPARQL ENGINES

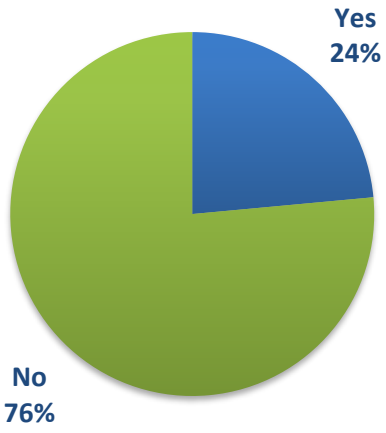
Result completeness?



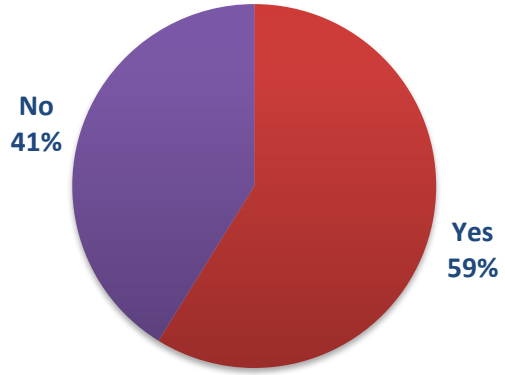
Policy-based planning?



Partial results retrieval?

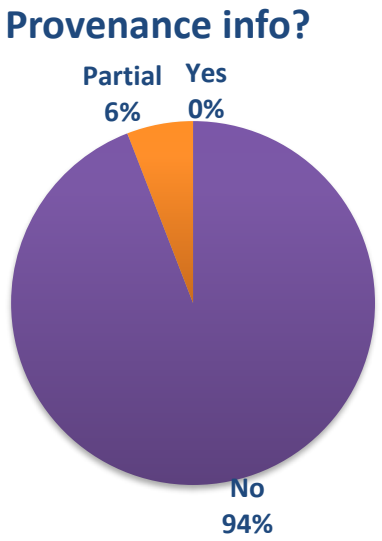


Adaptive processing?

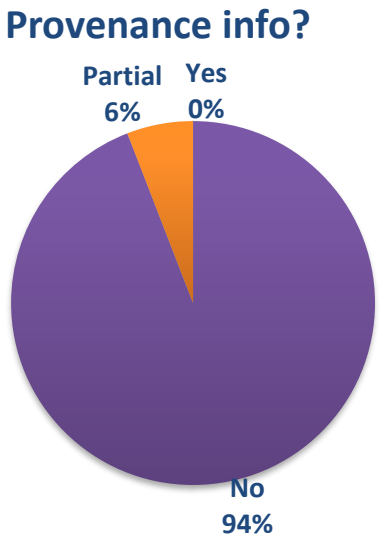


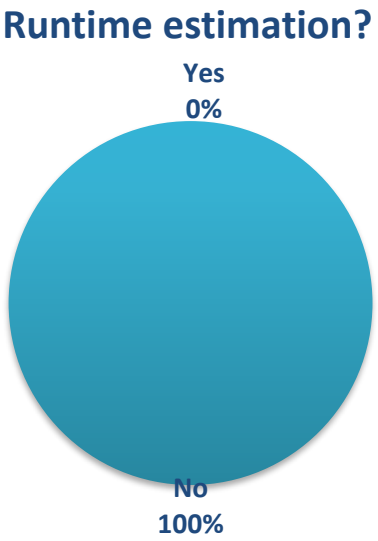
SURVEY: FEDERATED SPARQL ENGINES

SURVEY: FEDERATED SPARQL ENGINES



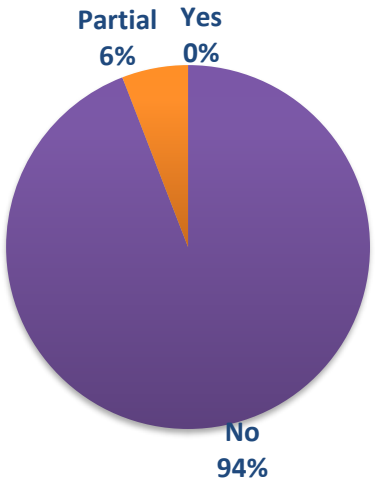
SURVEY: FEDERATED SPARQL ENGINES



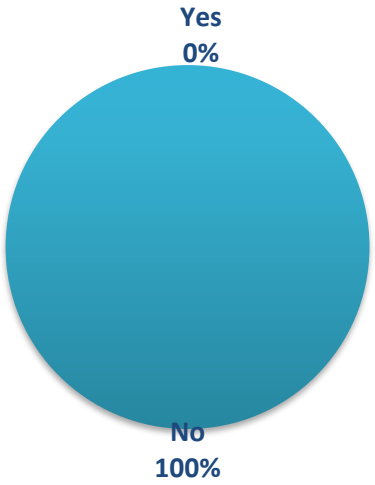


SURVEY: FEDERATED SPARQL ENGINES

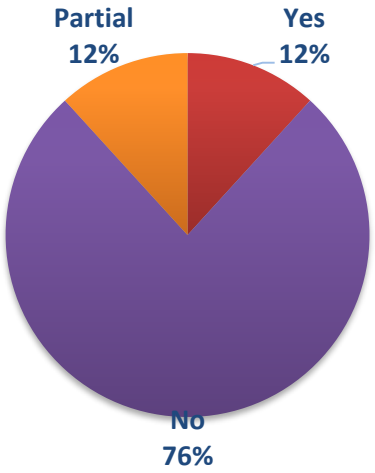
Provenance info?



Runtime estimation?

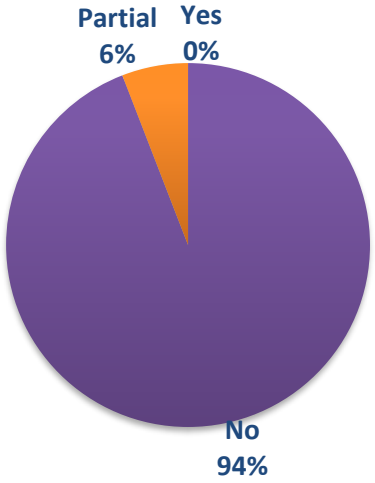


Duplicate detection?

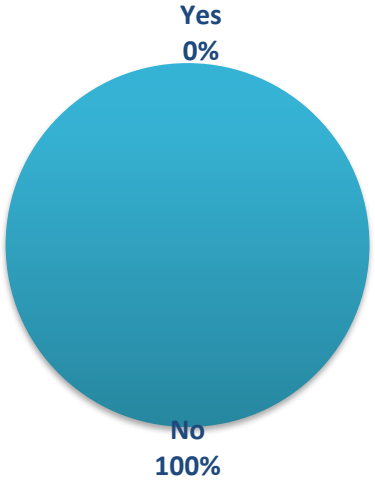


SURVEY: FEDERATED SPARQL ENGINES

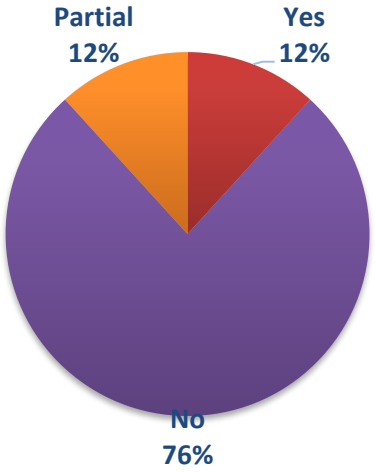
Provenance info?



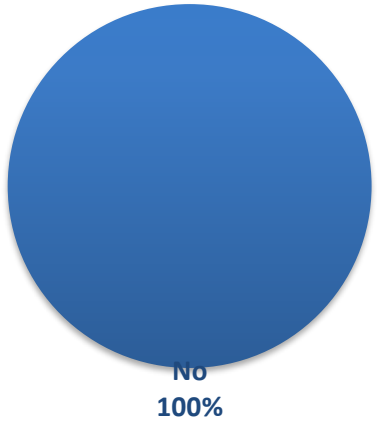
Runtime estimation?



Duplicate detection?



Top-k querying?



SURVEY: FEDERATED SPARQL ENGINES

SPL = SPLENDID, **FedS** = FedSearch, **GRA** = GRANATUM, **Ava** = Avalanche, **ANA** = ANAPSID, **ADE** = ADERIS

SPARQL Cluase	FedX	Atlas	LHD	SPL	FedS	GRA	Ava	DAW	LDQPS	SIHJoin	ANA	ADE	QWIDVD	DARQ	SAFE	QUETSAL
SERVICE	✓	✗	✗	✗	✓	✓	✓	✓	✗	✗	✓	✗	✓	✗	✓	✓
FILTER	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓
Unbound QP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Unbound QS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
OPTIONAL	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓	✓	✓
DISTINCT	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓
ORDER BY	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓	✓	✓
UNION	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓	✓	✓
NEGATION	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✓	✓	✓	✓
REGEX	✓	✗	✓	✗	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓
LIMIT	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓
CONSTRUCT	✓	✗	✓	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓
DESCRIBE	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓
ASK	✓	✗	✓	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓

PERFORMANCE VARIABLES

Independent Variables		Dependent/Observed Variables				
		#ASK	#TP Sources	Source Selection Time	Query Runtime	Answer Completeness
Query	query plan shape	✓	✓	✓	✓	✓
	#basic triple patterns	✓	✓	✓	✓	✓
	#instantiations and their position	✓	✓	✓	✓	x
	join selectivity	x	x	x	✓	x
	#intermediate results	x	x	x	✓	x
	answer size	x	x	x	✓	x
	usage of query language expressivity	✓	✓	✓	✓	x
	#general predicates	✓	✓	✓	✓	✓
Data	dataset size	x	x	x	✓	x
	data frequency distribution	x	x	x	✓	x
	type of partitioning	✓	✓	✓	✓	✓
	data endpoint distribution	✓	✓	✓	✓	✓
Platform	cache on/off	✓	✓	✓	✓	x
	RAM available	x	x	✓	✓	x
	#processors	x	x	✓	✓	x
Endpoints	#endpoints	✓	✓	✓	✓	✓
	endpoint type	x	x	✓	✓	x
	relation graph/endpoint/instance	x	x	x	✓	✓
	network latency	x	x	✓	✓	✓
	initial delay	x	x	✓	✓	x
	message size	x	x	x	✓	x
	transfer distribution	x	x	✓	✓	✓
	answer size limit	x	x	x	✓	✓
	timeout	x	x	x	✓	✓

EVALUATION

- Benchmarks
  - FedBench
  - Sliced FedBench
  - SP2Bench
- SPARQL endpoint federation engines
  - FedX
  - SPLENDID
  - LHD
  - DAW
  - ADERIS
  - ANAPSID

NUMBER OF ASK REQUESTS

			FedBench			
Query Category	FedX	SPLENDID	LHD	DARQ	ANAPSID	ADERIS
CD	252	44	0	0	43	0
LS	297	33	0	0	63	0
LD	369	19	0	0	37	0
Net	**918**	**96**	**0**	**0**	**143**	**0**
Query Category			**Sliced FedBench**			
CD	280	110	0	0	228	0
LS	330	100	0	0	143	0
LD	410	140	0	0	270	0
SP2Bench	660	180	0	0	265	0
Net	**1680**	**530**	**0**	**0**	**906**	**0**

TRIPLE PATTERN-WISE SOURCES SELECTED

	FedBench					
Query Category	FedX	SPLENDID	LHD	DARQ	ANAPSID	ADERIS
CD	78	78	112	84	36	84
LS	56	56	105	77	44	70
LD	108	108	119	112	54	119
Net	**242**	**242**	**336**	**283**	**134**	**273**
Query Category	**Sliced FedBench**					
CD	182	182	225	195	163	195
LS	125	125	163	133	102	117
LD	243	243	324	324	183	324
SP2Bench	521	521	593	420	244	173
Net	**1071**	**1071**	**1305**	**1072**	**692**	**809**

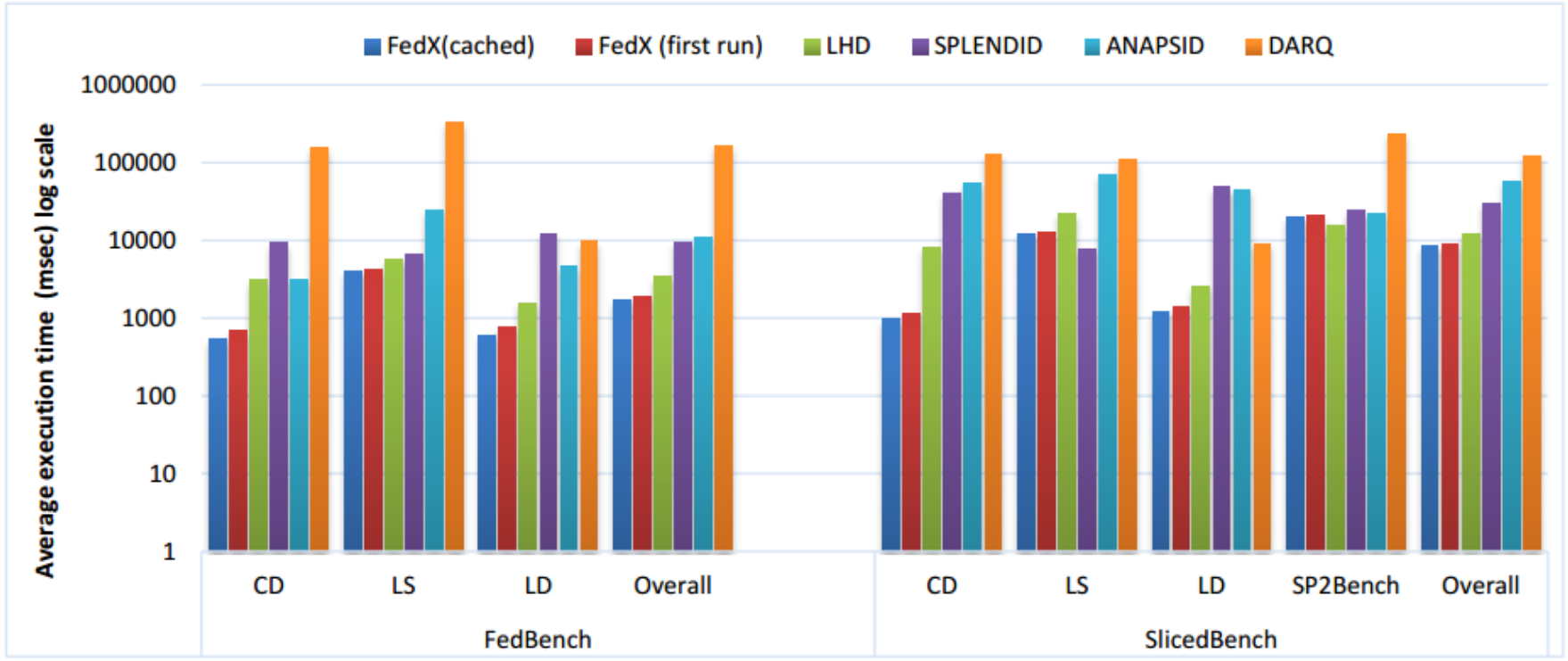
SOURCE SELECTION TIME (MS)

			FedBench				
Query Category	FedX(cold)	FedX(warm)	SPLENDID	LHD	DARQ	ANAPSID	ADERIS
CD	151	7	256	6	7	186	6
LS	147	8	236	6	12	477	5
LD	139	8	221	6	7	804	5
Average	**146**	**8**	**237**	**6**	**9**	**489**	**5**
Query Category			**Sliced FedBench**				
CD	284	9	470	7	8	3183	7
LS	207	7	308	7	8	2897	7
LD	323	9	557	8	7	2908	8
SP2Bench	212	9	738	8	9	RE	6
Average	**256**	**8**	**518**	**7**	**8**	**2996**	**7**

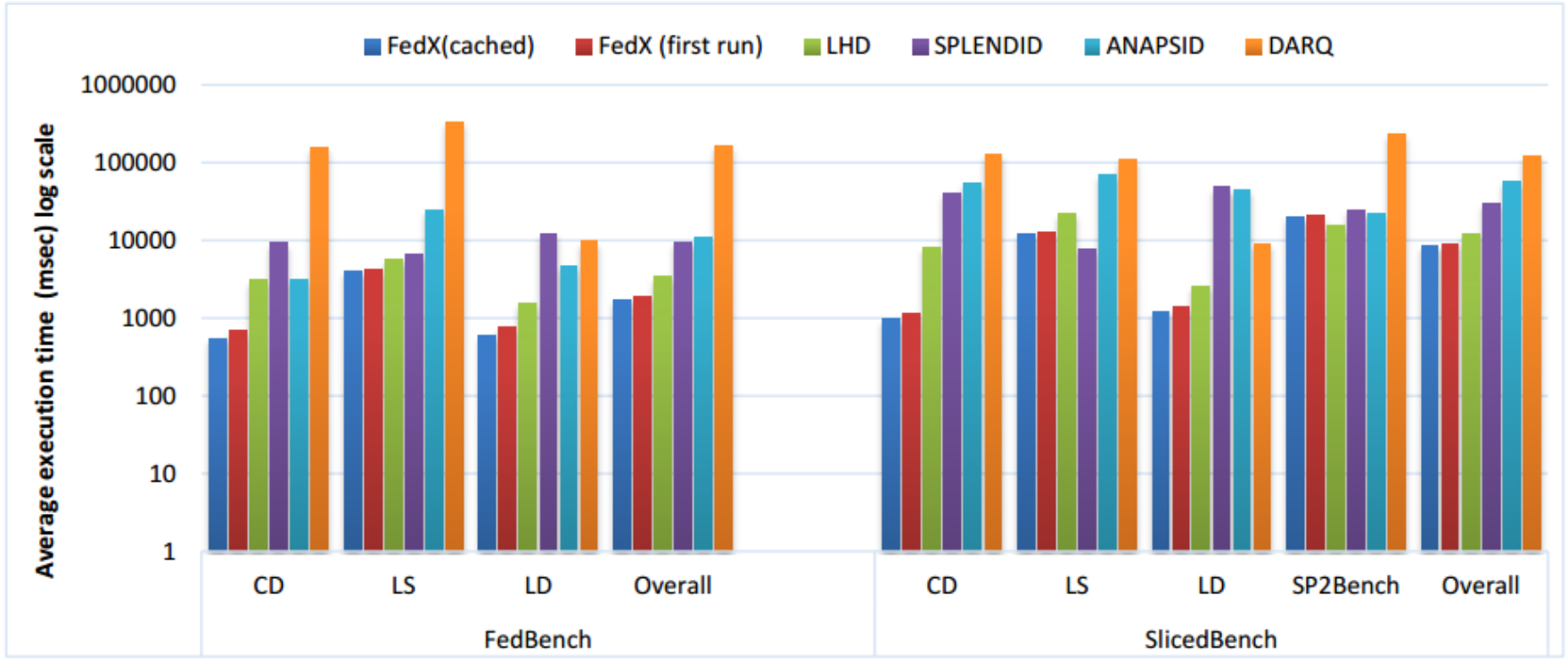
RESULT COMPLETENESS

	CD1(90)	CD7(1)	LS1(1159)	LS2(333)	LS3(9054)	LS5(393)	LD1(309)	LD3(162)	LD9(1)
SPLENDID	90	1	1159	333	9054	393	**308**	**159**	1
LHD	**77**	1	**0**	**322**	**0**	**0**	309	162	1
ANAPSID	90	**0**	1159	333	9054	393	309	162	1
ADERIS	**77**	1	1159	333	9054	393	309	162	1
DARQ	90	1	1159	333	9054	393	309	162	**0**
FedX	90	1	1159	333	9054	393	309	162	1

QUERY RUNTIME



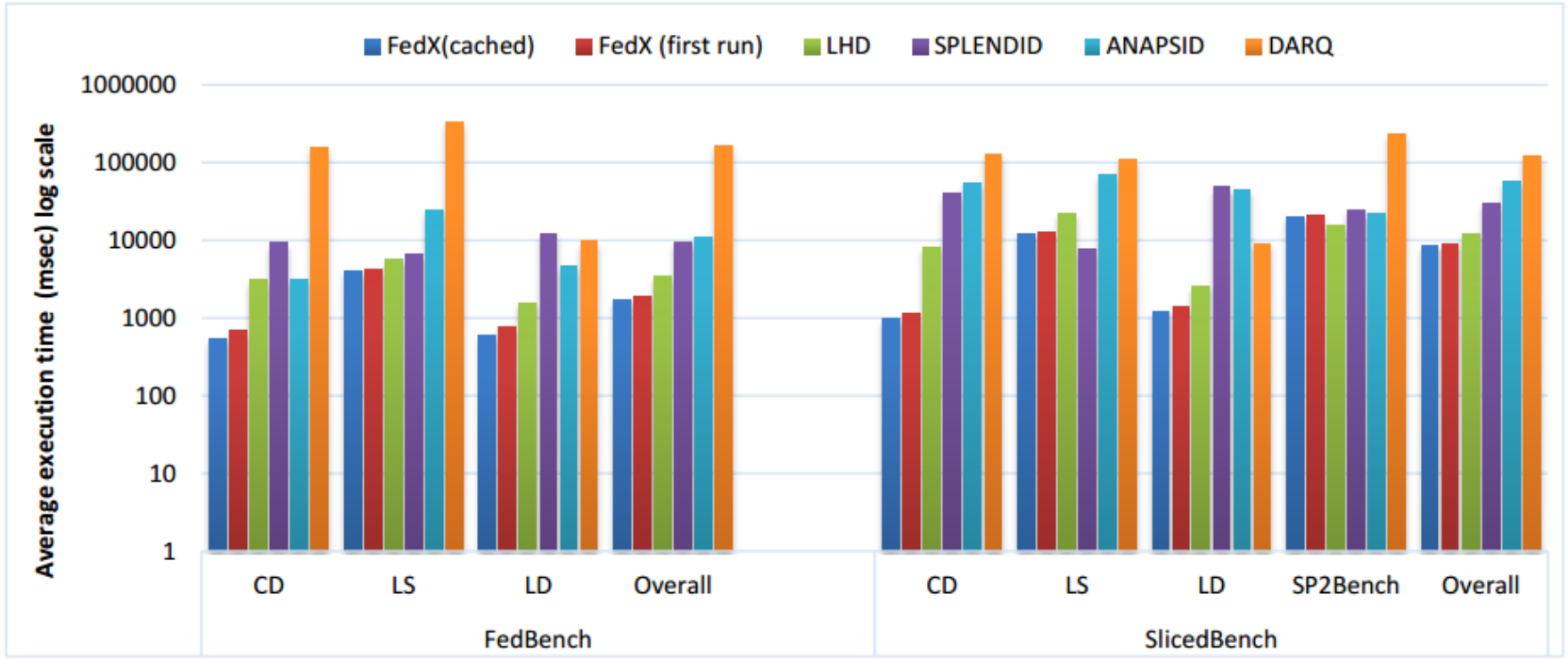
QUERY RUNTIME



25/25 17/22 13/22 15/24 16/22

FedBench: FedX(warm) → FedX(cold) → LHD → SPLENDID → ANAPSID → DARQ

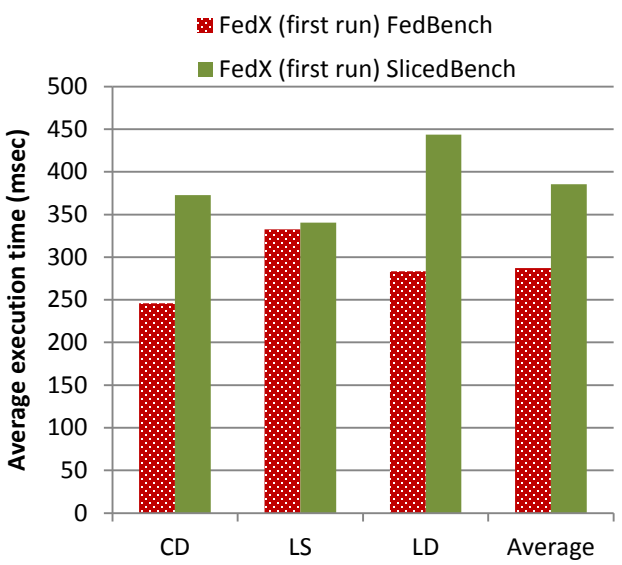
QUERY RUNTIME

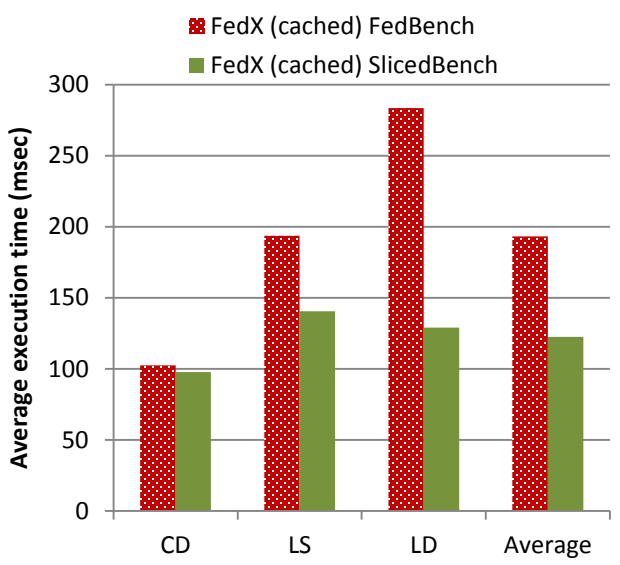


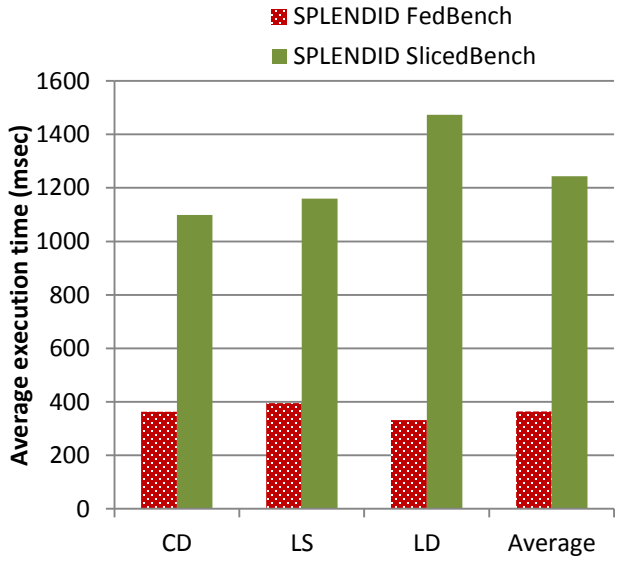
FedBench: FedX(warm) (25/25) → FedX(cold) (17/22) → LHD (13/22) → SPLENDID (15/24) → ANAPSID (16/22) → DARQ

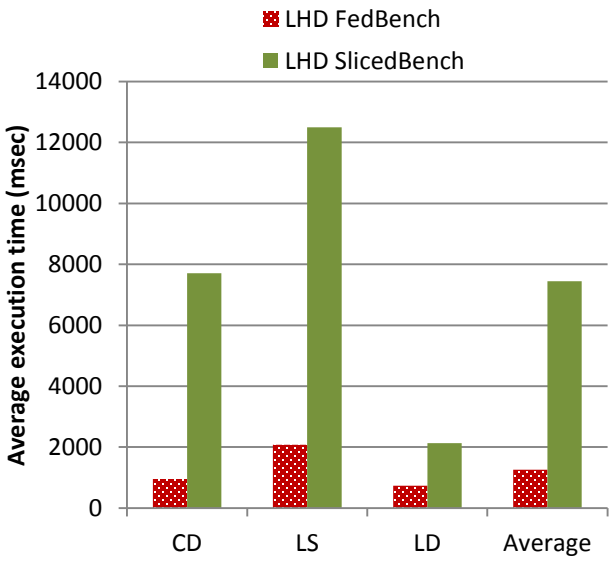
Sliced FedBench: FedX(warm) (29/36) → FedX(cold) (17/24) → LHD (17/24) → SPLENDID (17/26) → ANAPSID (12/20) → DARQ

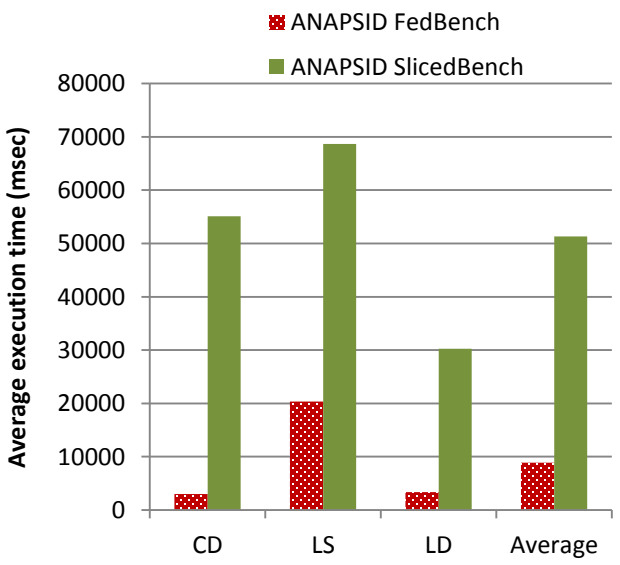
EFFECT OF DATA PARTITIONING

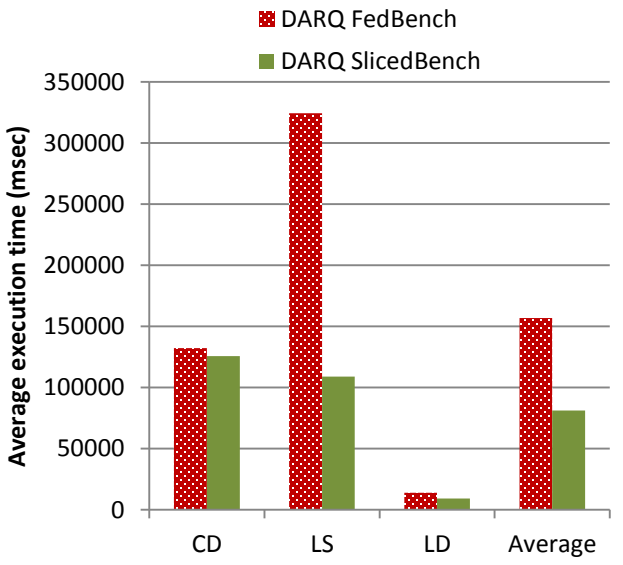




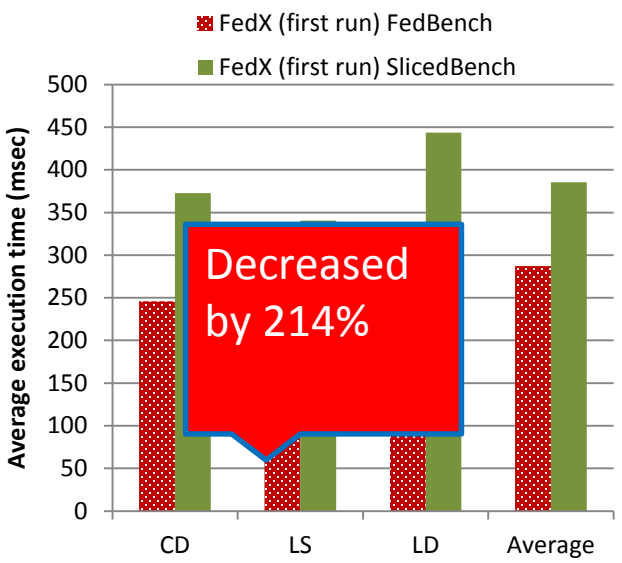


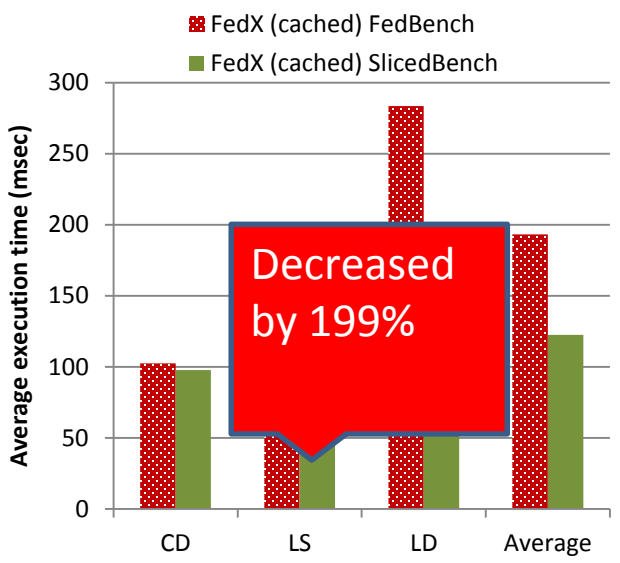


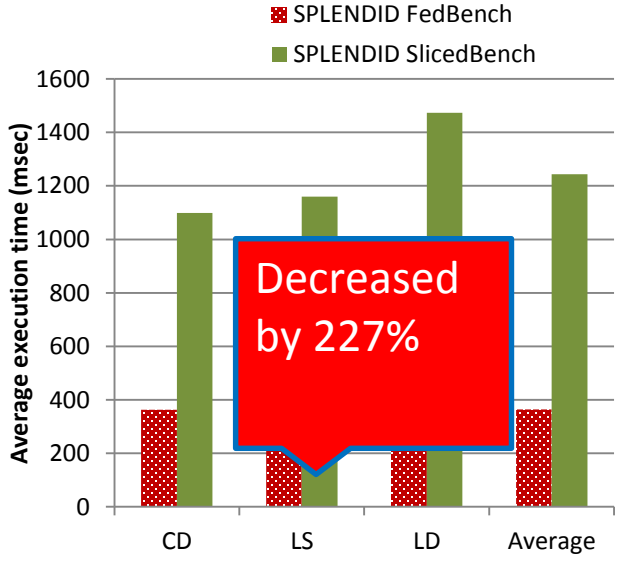


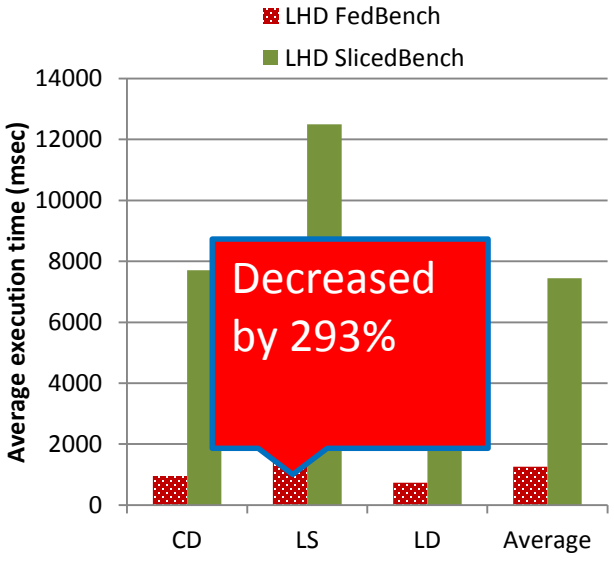


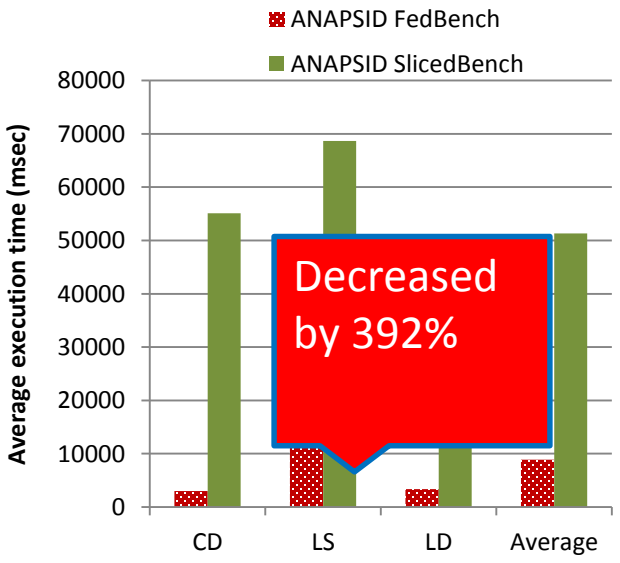
EFFECT OF DATA PARTITIONING

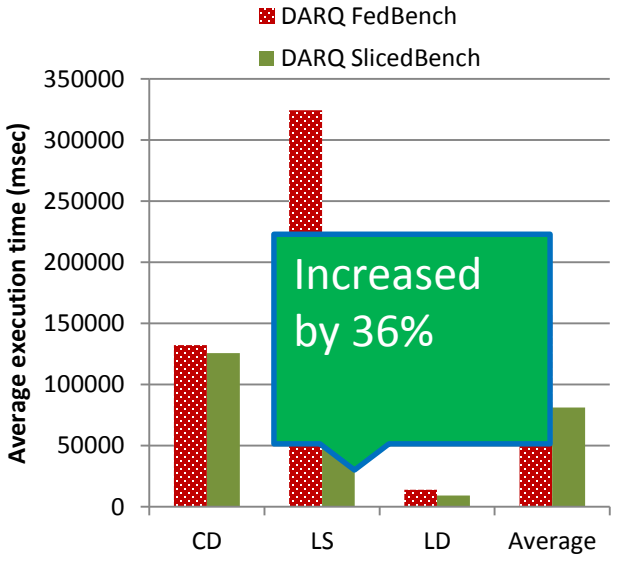












THANKS

{lastname}@informatik.uni-leipzig.de
AKSW, University of Leipzig, Germany