Interroger des Lacs de Données en utilisant Spark & Presto

Mohamed Nadjib Mami[†,§], Damien Graux[†,⋄], Simon Scerri[†,§], Hajira Jabeen[§], and Sören Auer[£*]
† Fraunhofer IAIS ; § University of Bonn
⋄ ADAPT Centre, Trinity College Dublin ; £ TIB and L3S Research Center
{mami,scerri,jabeen}@cs.uni-bonn.de;damien.graux@adaptcentre.ie;auer@l3s.de

ABSTRACT

Squerall est un outil permettant l'interrogation de sources de données hétérogènes à large échelle en utilisant à bon escient des moteurs de traitement dédiés aux larges volumes de données issus de la littérature: Spark et Presto. Les requêtes à destination des lacs de données sont évaluées à la volée, *i.e.*, directement sur les sources originelles sans procéder à de quelconques transformations préalables des données. Nous démontrons la capacité qu'a Squerall à interagir avec cinq sources différentes parmi lesquelles Cassandra et MongoDB. En particulier, nous mettons en évidence que notre outil peut joindre ensemble plusieurs sources en même temps, tout en montrant qu'étendre la couverture à d'autres sources potentielles reste simple. Des interfaces graphiques sont aussi mises à disposition pour (1) construire les requêtes SPARQL et (2) mettre en place les fichiers de configuration nécessaires.

CCS CONCEPTS

• **Information systems** → **Database query processing**; **Parallel and distributed DBMSs**; **Mediators and data integration**; • **Applied computing** → **Information integration and interoperability**; • **Computing methodologies** → *Knowledge representation and reasoning*.

1 INTRODUCTION

During the last four decades, a variety of data storage and management techniques have been developed in both research and industry. Today, we benefit from a multitude of storage solutions, varying in their data model (e.g. tabular, document, graph) or their ability to scale storage and querying. There are dozens of continuously evolving storage and data management solutions. As a result, users can choose a system that suits their individual application needs. However, those systems do not inter-operate, every stored datum is locked in the respective system it is stored in. For example, an e-commerce company might store *product* information in a Cassandra database, *offers* in MongoDB to benefit from its capability to store hierarchical multi-level values, and information about *Producers* obtained from an external source in a relational format. Without transforming and moving the data into a unified (scalable) data management solution, the data can hardly be explored and business insights be extracted using ad hoc uniform querying. We have taken on the mission of bridging this gap and developed Squerall[1] [9]: a software that gives access to heterogeneous data kept in their original forms and sources using Semantic Web techniques to enable uniform querying with SPARQL[2].

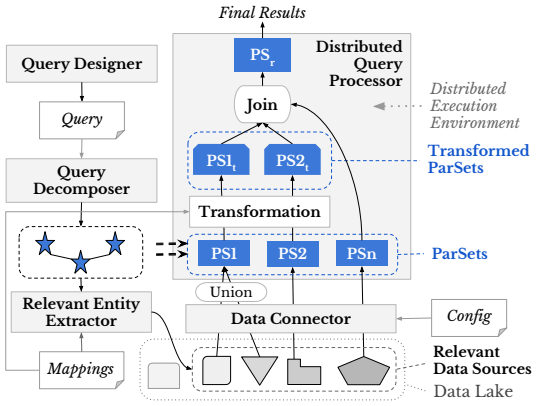

Figure 1: Squerall High-level Architecture.

Similar efforts to integrate and query large data sources exist in the literature. For instance, [3] defines a mapping language to express access links to NoSQL databases. [11] allows to run CRUD operations over NoSQL databases. [1] proposes a unifying *programming* model to directly access databases using *get*, *put* and *delete* primitives. [7] proposes a SQL-like language containing invocations to the native query interface of relational and NoSQL databases. [6] is a hybrid platform with consideration for both heterogeneous and dynamic data sources (streams). However, Squerall offers the highest number of supported data sources (namely: CSV, Parquet, Cassandra, MongoDB and MySQL) while providing the richest query capability, including joining, aggregation and ordering.

2 SQUERALL: CONCEPTS & ARCHITECTURE

Squerall [9] implements the so-called Ontology-Based Data Access (OBDA) [10] paradigm. In OBDA, data schemata are mapped to higher-level ontologies, forming a middleware against which queries are posed. These SPARQL queries are then executed in a separate distributed *environment*, which is, in particular, resilient to faults (node failure does not halt the entire query execution), and elastic and horizontally-scalable (more nodes can be added to accommodate more expensive computations). In addition, as data from different sources is generated by different applications, they

*This research was partially supported by the European Union's H2020 research and innovation programme BETTER (GA. 776280); and by the ADAPT Centre for Digital Content Technology funded under the SFI Research Centres Programme (Grant 13/RC/2106) and co-funded under the European Regional Development Fund.

[1] Associated website: <https://eis-bonn.github.io/Squerall/>
[2] <http://www.w3.org/TR/sparql11-overview/>

	SPARQL SELECT Queries adapted from BSBM								
	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q10**
– Scale 1: scale factor of 0.5 millions products –									
Presto	55.34	28.89	15.84	53.63	49.24	43.38	18.63	14.38	89.08
Spark	98.78	189.57	59.96	277.30	222.76	191.26	159.51	91.38	300.38
Diff. %	**178.48**	**656.17**	**378.57**	**517.06**	**452.40**	**440.88**	**856.31**	**635.33**	**337.21**
– Scale 2: scale factor of 1.5 millions products –									
Presto	139.15	48.59	16.30	133.45	115.35	116.08	42.16	14.37	405.84
Spark	102.86	584.67	70.76	673.12	637.18	611.65	447.27	75.19	888.98
Diff. %	73.92	**1203.36**	**434.05**	**504.41**	**552.40**	**526.93**	**1060.83**	**523.31**	**219.05**
– Scale 3: scale factor of 5 millions products –									
Presto	276.91	131.87	30.58	340.61	350.04	334.29	98.11	18.96	784.01
Spark	132.37	1813.69	93.19	2131.10	1846.59	1833.47	1390.99	79.33	2703.43
Diff. %	47.80	**1375.40**	**304.71**	**625.67**	**527.54**	**548.46**	**1417.80**	**418.47**	**344.82**

Table 1: Query execution times (seconds) using Presto and Spark and the difference percentage between them (%).

may not be able to be readily cross-joined. Thus, modifications on the possible join values ought to be incorporated. `Squerall` is comprised of five components (see Figure 1):

- **Query Decomposer:** Validates and analyzes SPARQL queries provided by a user. Particularly, the Query Decomposer extracts the Basic Graph Pattern fragment of the query and decomposes it into star-shaped sub-graph patterns having the same subject, *stars* for short. This component also detects links between stars.
- **Relevant Entity Extractor:** Each star is analyzed separately; this component searches in the mappings for entities that are mapped to every predicate of the star.
- **Data Connector:** Once relevant data entities are detected, they are connected to the distributed execution environment. Every detected entity is loaded into a *ParSet* (Parallel dataSet): a data structure that can be distributed and operated on concurrently. The Data Connector expects users to input connection metadata.
- **Distributed Query Processor:** Following the principles introduced earlier, queries are executed in parallel. Query execution occurs on and across the ParSets. Links between stars retrieved by the Query Decomposer are transformed into joins between the relevant detected data entities and all stars are incrementally joined. When disjoinability points are known, join values are altered to enable joinability.
- **Query Designer:** We see the necessity of supporting users in their SPARQL query creation.

3 TECHNICAL DETAILS

Core technologies. RML [5] and FNO [4] are used to express mappings and to declare query-independent transformations. *Apache Spark* and *Presto* are used to implement both the Data Connector and Distributed Query Processor. Spark is a general-purpose processing engine, and Presto is a distributed SQL query engine. Both provide dozens of wrappers[3] to connect to a data source. They load the data (fully, or partially) to their in-memory data structures.

Achieved Performance. We evaluate the performance querying five different data sources: Cassandra, MySQL, MongoDB, Parquet, and CSV. As evaluation data we choose the BSBM [2] benchmark. We pick five relational tables (*Product, Producer, Offer, Review, and Person*) and load them into the five data sources. For performance and scalability we evaluate the execution time of BSBM queries against three increasing data sizes: 0.5m, 1,5m and 5m scale factors (number of products). We experiment with all BSBM `SELECT` queries with some adaptation (queries are altered to involve the five tables we populated). The evaluation was carried out in a cluster of three nodes, each having a 16-core DELL PowerEdge processor, 256GB RAM, and 3TB SATA disk. Presto-based `Squerall` exhibits better performance to the Spark-based alternative in the majority of the cases. To measure this difference, we calculate the difference, in percentage, between Presto and Spark execution times (third row under each scale result in Table 1), e.g., for Q2 in the first scale Presto-based `Squerall` is 656% faster.

User Interfaces. We provide 3 GUIs to help users produce `Squerall`'s inputs: *Config*, *Mappings* files and *SPARQL query*. They have built-in search functionalities that send requests to the LOV catalog[4] to search for adequate terms from existing ontologies.

4 CONCLUSION

`Squerall` addresses the *Variety* challenge of Big Data by making use of Semantic Web standards and best practices. It can be extended to embrace new data sources, by making use of the query engines' own wrappers. Additionally, `Squerall` has been integrated[5] into SANSA [8], a framework for scalable processing and analysis of large-scale RDF data, widening its scope to also access non-RDF data sources. `Squerall` source code is available under an Apache-2.0 license on GitHub[6]. In addition, a screencast presenting the various interfaces and the query execution is available[7]. The deployment is further facilitated with a Dockerfile to quickly run the BSBM use-case described here.

REFERENCES

[1] Paolo Atzeni, Francesca Bugiotti, and Luca Rossi. 2012. Uniform access to non-relational database systems: The SOS platform. In *International Conference on Advanced Information Systems Engineering*. Springer, 160–174.

[2] Christian Bizer and Andreas Schultz. 2009. The berlin SPARQL benchmark. *International Journal on Semantic Web and Information Systems* 5, 2 (2009), 1–24.

[3] Olivier Curé, Robin Hecht, Chan Le Duc, and Myriam Lamolle. 2011. Data integration over nosql stores using access path based mappings. In *International Conference on Database and Expert Systems Applications*. Springer, 481–495.

[4] Ben De Meester, Wouter Maroy, Anastasia Dimou, Ruben Verborgh, and Erik Mannens. 2017. Declarative data transformations for Linked Data generation: the case of DBpedia. In *European Semantic Web Conference*. Springer, 33–48.

[5] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. 2014. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data.. In *LDOW*.

[6] Martin Giese, Ahmet Soylu, Guillermo Vega-Gorgojo, Arild Waaler, Peter Haase, Ernesto Jiménez-Ruiz, Davide Lanti, Martín Rezk, Guohui Xiao, Özgür Özçep, et al. 2015. Optique: Zooming in on big data. *Computer* 48, 3 (2015), 60–67.

[7] Boyan Kolev, Patrick Valduriez, Carlyna Bondiombouy, Ricardo Jimenez-Peris, Raquel Pau, and José Pereira. 2016. CloudMdsQL: querying heterogeneous cloud data stores with a common language. *Distributed and parallel databases* 34, 4 (2016), 463–503.

[8] Jens Lehmann, Gezim Sejdiu, Lorenz Bühmann, Patrick Westphal, Claus Stadler, Ivan Ermilov, Simon Bin, Nilesh Chakraborty, Muhammad Saleem, Axel-Cyrille Ngonga Ngomo, and Hajira Jabeen. 2017. Distributed Semantic Analytics using the SANSA Stack. In *ISWC Resources Track*.

[9] Mohamed Nadjib Mami, Damien Graux, Simon Scerri, Hajira Jabeen, Sören Auer, and Jens Lehmann. 2019. Squerall: Virtual ontology-based access to heterogeneous and large data sources. In *International Semantic Web Conference*. Springer, 229–245.

[10] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking data to ontologies. In *Journal on Data Semantics X*. Springer, 133–173.

[11] Rami Sellami and Bruno Defude. 2018. Complex queries optimization and evaluation over relational and NoSQL data stores in cloud environments. *IEEE Transactions on Big Data* 4, 2 (2018), 217–230.

[3] <https://spark-packages.org> and <https://prestodb.io/docs/current/connector.html>

[4] Linked Open Vocabularies: publish and search ontologies <https://lov.linkeddata.es/>

[5] <https://github.com/SANSA-Stack/SANSA-DataLake>

[6] <https://github.com/EIS-Bonn/Squerall>

[7] <https://github.com/EIS-Bonn/Squerall/blob/master/evaluation/screencasts>