Benchmarking RDF Stores for Smart City Services

Pierfrancesco Bellini, Paolo Nesi, Gianni Pantaleo
Distributed Systems and Internet Technology Lab, DISIT, http://www.disit.org
Department of Information Engineering, DINFO, University of Florence, Florence, Italy
pierfrancesco.bellini@unifi.it, paolo.nesi@unifi.it , gianni.pantaleo@unifi.it

***Abstract*— Smart cities are providing advanced services gathering data from different sources. Cities collect static data like road graphs, service description as well as dynamic/real time data like weather forecast, traffic sensors, bus positions, events, emergency data, etc. RDF stores may be used to integrate all information coming from different sources and allow applications to use the data to provide new advanced services to the citizens and city administrators exploiting inferential capabilities. These city services are typically based on geographic positions and need to access quickly to the real time data (e.g., next time of bus arrival) as well as to the historical data to perform some data analysis to compute predictions. In this paper, the needs and constraints for RDF stores to be used for smart cities services and the currently available RDF stores are evaluated. The assessment model allows understanding if they are suitable as a basis for Smart City modeling and application. The benchmark proposed has been defined for generic smart city services to compare results that can be obtained using different RDF Stores. In the benchmark, particular emphasis is devoted to geo and full text searches that are partially considered in other well-known RDF store benchmarks as LUBM and BSBM. The paper reports the validation of the proposed *Smart City RDF Benchmark* (http://www.disit.org/smartcityrdfbenchmark) on the basis of Florence Smart City accessible as Km4City. The comparison addressed a number of well-known RDF stores as Virtuoso, GraphDB and many others.**

Keywords*— *smart city; RDF stores; graph databases; RDF benchmark; linked data benchmark.

I. INTRODUCTION

Smart cities produce large amount of data having a large variability, variety, velocity, and size; and thus complexity. The variety and variability of data can be due to the presence of several different formats [1] and to the interoperability among semantics of the single fields and of the several data sets [2]. The data velocity is related to the frequency of data update, and allows distinguishing static from dynamic data. Static data are rarely updated, such as once per month/year, as opposed to the dynamic data which are updated: from once a day up to every minute to arrive at real time data. The size grows over time accumulating new data every day and week. The usage of Resource Description Framework (RDF) stores in the application domain of Smart City is quite recent, since in most cases the services are vertically provided. For example the Intelligent Transport System, ITS, in the city only provides information regarding the location of buses and their delay, without addressing the location of services or real time events in the city, and associated with the bus stops. The integrated services are typically provided by aggregators that exploit data integration models. Some city data integrators are well-known services such as bike and car sharing, navigator system, tourism information, hotel booking, etc. All these solutions have the need to integrate geo-located information with real time data and events continuously arriving from updated information such as: events, votes, traffic flows, comments, etc. [4]. For these applications, RDF stores may be a solution to allow addressing the variability of data, to make reasoning on space, time, and concepts. On this regard, a comprehensive smart city data model and ontology can be really effective (see for example http://smartcity.linkeddata.es/), produced by Read4SmartCity project http://www.ready4smartcities.eu/ of the European Commission. One of the most ranked models in the Read4SmartCity research project is Km4City that is also used as a reference for the definition of the data of the proposed benchmark [2].

For the evaluation of RDF stores, some benchmarks were developed. Some of them are based on real-world datasets while others provide a program to generate a synthetic dataset. For example, the LUBM benchmark [3] uses a synthetic dataset in the university domain; however it covers only the SPARQL 1.0 specification. Another example is the BSBM benchmark [5] that generates a synthetic dataset in the e-commerce domain, and covers the SPARQL 1.1 business analytics queries. More recently, in the Linked Data Benchmarks Council project (http://ldbcouncil.org) two benchmarks were developed both generating a synthetic dataset, one from the semantic publishing domain (LDBC-SP) and the other from the social networks domain (LDBC-SN). In these benchmarks not only query performance but a mix of insert/update/delete and query operations are considered. It should be noticed that the current SPARQL specification does not cover the spatial and keyword searches thus a query involving these aspects needs to be adapted. The GeoSPARQL standard [6] was developed by the Open Geospatial Consortium to overcome this problem, while not many solutions currently support this specification. Regarding the benchmark of spatial RDF stores the geographic benchmark [7] was developed by using both a synthetic generated dataset and a real dataset. It aims to analyze the support and performance for advanced spatial relationships among complex spatial entities (e.g., polygons).

In this paper, the needs and constraints for RDF stores to be used for smart cities services and the currently available RDF stores are evaluated. The assessment model allows understanding if they are suitable as a basis for Smart City modeling and application. Moreover, a benchmark for linked data, RDF / Graph data base / stores with a special care to the real structures and relationships that may be present in smart city applications is presented. The benchmark proposed has

been defined for smart city services to compare results that can be obtained by using different RDF Stores. In the benchmark, particular emphasis is devoted to geo and full text searches that are partially considered in other benchmarks. The research described in this paper reports the validation of the proposed *Smart City RDF Benchmark* on the basis of Florence Smart City grounded on Km4City ontology and model (web page to access at the benchmark data and queries http://www.disit.org/smartcityrdfbenchmark). The comparison addressed a number of well-known RDF stores as Virtuoso, GraphDB and many others.

The paper is structured as follows. In Section II, the major smart city requirements/demands are reported. Section III presents the general evaluation methodology for assessing and selecting the RDF stores for smart city applications. In Section IV, the details on the proposed *Smart City RDF Benchmark* are reported. In Section V, the comparison of most relevant state of the art RDF stores under evaluation is reported on the basis of the model identified in Section III. Section VI reports the application of the proposed benchmark in assessing the most interesting RDF stores (Virtuoso and GraphDB). Conclusions are drawn in Section VII.

II. SMART CITY REQUIREMENTS FOR RDF STORES

When providing services to citizens of a smart city an RDF/graph store should have some features that allow supporting functionalities. RDF stores must support (i) *spatial indexing* to allow to quickly provide information near to a given geographical GPS point, it should also support advanced geo-spatial functionalities as being able to manage complex geometries (e.g., a cycle path, a parking area); (ii) *full text indexing* allowing integrating keyword search with more advanced semantic queries; (iii) handle *quadruples* (not only triples) to allow associating dataset metadata with the triples loaded, since data come from many different sources and it is important to track the data provenience, metadata and the associated license; (iv) *some kind of inference* like the basic RDFS or the more advanced OWL2 profiles allowing inferring new facts from the data available; (v) *temporal indexing*, since many information and features are changing over time in smart cities (e.g., weather situation and forecast, traffic flow, bus position, events happening in the city), for this reason, it is quite important that the RDF store should support temporal search to allow the easy retrieval of temporal data; (vi) *high volume of queries*, handling big RDF datasets with many users querying the data are quite challenging, for this reason a clustering solution is needed.

Another important point is that the solution should be open source to avoid the risk of technology lock-in especially for very new technologies as the RDF stores. Moreover there should be an active community handling and supporting the product.

III. EVALUATION METHODOLOGY

The evaluation methodology is performed in two phases. In the first phase, an analysis of general features has been performed by following the requirements provided in Section II. In the second phase, performance tests have been designed by using three datasets with growing size expanding temporal horizon (1 month, 2 months and 3 months of real time data cumulated). On these datasets, specific queries have been designed by considering, all the aspects, and including spatial and full text searches. The tests for assessing performance have been performed on stores supporting the set of features considered basilar, as reported in the following.

Therefore, the features taken into account to analyse the RDF stores have been: (i) The *SPARQL version* supported being 1.0 or 1.1; (ii) the inference type supported as full materialization of triples at load time or materialization at query time, and the inference profiles supported (e.g., RDFS, RDFS+, OWL, OWL2, …); (iii) if the store is a triple or quadruple store, that store only the *subject predicate object* or it can have also a *context* URI; (iv) how the triples/quadruples are physically stored like using a custom indexing or an RDBMS or other external service (e.g., HBase, Cassandra); (v) if the store supports *Horizontal Clustering* where replicated nodes are used for a high availability and fault tolerant solution; (vi) if the store supports *Vertical Clustering* where data are replicated on multiple nodes while no node contains all the data (index sharing); (vii) If the store supports *Spatial search* as Basic (meaning that it is able to index and retrieve only geolocated points) or Advanced (meaning that it is able to index complex shapes, for example polylines); (viii) if the store supports full text search, providing the ability to search using keywords; (ix) if the store allows associating triple/quadruples with a temporal validity context allowing to easily filter triples using temporal constraints; (x) the size of stores managed as the largest number of triples/quadruples reported to be managed by the RDF store in the literature; (xi) the License under which the RDF store is available, being it open source or commercial; (xii) the development language (e.g., Java, C); (xiii) if the project is still active.

Detailed performance testing should be performed on stores that support the following minimum set of requirements: (i) supporting SPARQL 1.1 as it provides aggregation functions (group by, count) and other features that were missing in 1.0; (ii) support for at least RDFS inference at load time or query time; (iii) support for quadruple stores so that provenience metadata can be associated with datasets; (iv) support for at least basic spatial search to allow searching services via geographical position; (v) support for full text search to be able to integrate keyword search with semantic search; (vi) supporting "Big stores". If the store supports additional features they are positively considered.

IV. SMART CITY RDF BENCHMARK

In this section, the main elements of the *Smart City RDF Benchmark* are presented. First the datasets used are described, and then the queries to be used for the assessment of performances are motivated.

A. Dataset of the Smart City RDF Benchmark

The data used for the evaluation is based on the Km4City knowledge base [2]. The Km4City models many aspects of a smart city. Some of them are static (or quasi-static) data such as (i) the *road graph* modeling the roads, the public administrations, etc. (ii) the "services" that are present in the city (e.g., restaurants, hotels, cycle paths, …) that are associated with the road graph and organized in an hierarchy, (iii) the bus stops, bus lines of the local transportation, (iv) the

road sensors that are present on the roads. Moreover, dynamic information that change over time is also modelled, such as: (i) the weather forecasts for the different municipalities, (ii) the status/position of the bus with eventual forecasts for the arrival at the bus stops, (iii) the status of the parking lots (e.g., number of free places), (iv) the readings of the traffic sensors, (v) the events defined on the city. The testing datasets, comprised of triples, have been generated on the basis of Km4City model by using data from the Florence smart city service.

Three different datasets has been generated. They share the same 'static' information and differ for the dynamic part, having one, two or three months of history, respectively, in the past of the dynamic information. In Table I, the numbers of quadruples that are present for the different parts of the Km4City ontology are reported.

TABLE I. DATASET DISTRIBUTION

Type	1 month		2 months		3 months	
	quadruples	*%*	*quadruples*	*%*	*quadruples*	*%*
AVM	8.4M	19%	18M	33%	28M	43.1%
Parking	413k	0.9%	976k	1.8%	1.4M	2.1%
Sensors	900k	2%	1.7M	3.1%	2.2M	3.3%
Meteo	15k	0%	23k	0%	23k	0%
Total dynamic	**9.7M**	**22%**	**21M**	**38%**	**32.5M**	**48.5%**
Road graph	33.5M	75%	33.5M	60.3%	33.5M	50%
Services	681k	1.5%	681k	1.2%	681k	1%
Other static	286k	0.6%	286k	0.5%	286k	0.4%
Total static	**34.5M**	**78%**	**34.5M**	**62%**	**34.5M**	**51.4%**
Total	**44.2M**	**100%**	**55.6M**	**100%**	**67.5M**	**100%**

B. *SPARQL Queries of the Smart City RDF Benchmark*

The queries performed over the dataset are mainly those used in http://servicemap.disit.org, and thus a live solution can be accessed. It should be noted that the SPARQL recommendation [10] does not cover the geo-spatial queries and neither the full-text queries. Therefore, in order to support those features, RDF store builder/vendor implemented the feature with a specific syntax. For this reason for some queries there is not a unique formulation and the query has to be adapted for each RDF store under test (they can be accessed at http://www.disit.org/smartcityrdfbenchmark). In the benchmark there are 25 queries and 8 of them use inferred information.

V. COMPARISON OF RDF STORES

In this section, the RDF stores under assessment are compared on the basis of the feature model identified and discussed in Section III. The comparison is performed with the aim of identifying those that are better ranked to be used on smart city applications. The RDF store solutions that support all the minimum requirements are Virtuoso 7.2.0.1 open source and commercial edition and GraphDB Standard Edition 6.1. As a consequence, only the Virtuoso open source edition and GraphDB SE have been assessed in term of performance, as reported in Section VI. The RDF stores considered in the assessment are briefly described in the following.

Virtuoso 7.2.0.1 [9], is a SPARQL 1.1 quadruple store developed in C available both with open source and commercial license. The open source version mainly misses the horizontal clustering feature. Inference is not materialized at load time, while query rewrite is performed to support RDFS+ inference. It is backed by the Virtuoso RDBMS and SPARQL queries are translated to SQL. It supports advanced spatial indexing and supports full text search. The community behind virtuoso is leaded by OpenLink Software ltd and it is quite active.

GraphDB SE 6.1 (former OWLIM store) (http://ontotext.com/products/ontotext-graphdb/) is a commercial solution providing a SPARQL 1.1 endpoint supporting triple/quadruple stores with spatial indexing of geographic coordinates and full text indexing based on Lucene. It supports inference at load time with different rule sets (RDFS, OWL2RL, etc.), and rule sets can be selected by the user. It supports up to 10 billion of triples on a single node. The Enterprise edition allows horizontal scaling. The solution is implemented in Java using OpenRDF Sesame. The project is still active and it is managed by Ontotext.

Blazegraph (ex BigData) (https://wiki.blazegraph.com) is an open source project with also a commercial license. It supports triple and quadruple stores. RDFS+ inference (at load time) is available only on triple stores. It has a full-text indexing support, and not geospatial indexing. It provides both a horizontal and vertical scaling solution allowing an index to be shared on multiple nodes. A single computer can manage up to 50 billion triples. The project is managed by Systap and it is still active.

CumulusRDF [10] is an open source project based on OpenRDF Sesame using Apache Cassandra 1.2 as NoSQL storage layer. It does not support inference and can store only triples. Since it is based on Cassandra, it supports vertical scaling for storage of the indexes on the nodes in the cluster, while only one node is used to perform queries.

Stardog v3 (http://stardog.com/) is a commercial RDF quadruple store developed by Clark&Parsia. It supports SPARQL 1.1 and OWL2 inference at query time, full-text indexing and search, while the spatial indexing is not supported. It allows horizontal scaling, and it is a quite active project. Stardog supports 10 billion triples on single node.

Strabon [11] is an open source SPARQL 1.1 store developed to support both spatial and temporal search. It is based on PostGIS extension of Postgres RDBMS; it does not support inference, and neither full-text search. It only provides support for storing triples (the context URI associated with the triple is used for temporal linking). No clustering solution is available.

VI. PERFORMANCE ASSESSMENT

The performance evaluation has been carried out by considering the data load time and the query execution time, for space limitations the full results are available on http://www.disit.org/smartcityrdfbenchmark and in the following is reported a synthesis. For loading 67M quadruples of the 3 months dataset Graph DB took about 8h and 12m while virtuoso 3h and 22m. GraphDB is about three times slower than Virtuoso due to the fact that GraphDB performs

inference at load time while Virtuoso at query time. And also the number of triples indexed in GraphDB (106M) is 36% bigger than those of Virtuoso (69M). For Virtuoso, the increment of triples stored with respect to those stated (2.1M for the 3 months case) is only due to transform the geo:lat and geo:long triples in a geo:geometry with POINT() to enable the geo-spatial indexing. While in the same case, for GraphDB the increment of 39M triples is due to the materialization of inference. The queries were performed with the three datasets and were tested performing a pseudo-random sequence of 500 queries repeated two times with some pseudo-random arguments in order to reduce the caching effect. The sequence of queries performed is the same for each execution in order to test the same sequence on different systems. From the query results, when no spatial and full text search and inference are involved, the performance is quite comparable, and in some cases GraphDB is better ranked. When inference is needed in the case of Virtuoso the inference should be enabled on the single constraint involving a general class (e.g., all services in the Accommodation class). While if the inference is enabled, generally on the query, the internal automated query rewrite takes a very long time (perhaps due to the size of the ontologies used). For example, for query *Service-Acc-Clt-Trs-W&F-florence* that search for services in Florence that are Accommodations, Cultural activities, Transport or Wine and Food, in Virtuoso the time grows from an average of 2.62s to an average of 24.5s (on the 3 months dataset) while GraphDB takes about 11.45s. When considering the spatial indexing we found in Virtuoso various problems using the *st_intersection* function. In some cases, Virtuoso returns an error, in other cases providing a lower number of results with respect to the correctly expected and providing different results for the same query on the three different datasets that do not differ for the part considered in the query. On the other hand, in Virtuoso, if the *st_distance* function is used, all the obtained results have been verified to be correct, apart from few cases on the border (due to the numerical computation in measuring distances). The usage of the distance function for Virtuoso is good solution in most cases for example query retrieving all services within 5km from a gps position on the 3 months datasets takes 1.5s on virtuoso using st_distance function while it takes 9.7s on GraphDB, but reducing the distance to 200m Virtuoso takes 248ms while GraphDB only 153ms. Using the st_distance function on Virtuoso seems that the query optimizer to do not exploit the spatial index. This fact may be deduced from comparing that a same query (*Find-address*) by using *st_distance* function takes about 6s while using the *st_intersect* function takes about 0.3s. Another aspect to be considered is the mixing of spatial query with text search query. With GraphDB, we registered very long execution time hitting in some cases the timeout of one hour. In this case of mixing spatial and text search for Virtuoso, the queries using st_intersect function returned an error while using the st_distance function takes only 157 ms. Regarding the analytic queries that count the daily number of records of the weather forecasts, bus, sensor data, parking status for the three datasets Virtuoso is better ranked, it has an execution time less than 404ms while GraphDB is less than 3s. Moreover Virtuoso presents a lower growing factor with respect to GraphDB.

VII. CONCLUSION

In this paper we have reported a comparative study about state of the art RDF stores on the basis of their main features and in particular on the SPARQL aspects. In addition, the *Smart City RDF Benchmark* has been proposed. The benchmark is based on (i) some datasets of quadruples (grounded on Km4City model); (ii) a set of SPARQL queries. In the benchmark, particular emphasis is devoted to geo and full text searches that have been partially considered in state of the art. The comparison addressed a number of well-known RDF stores, and in particular Virtuoso and GraphDB for the performance aspects. As a general consideration, regarding the performance, it should be noted that Virtuoso performs better when decreasing the selectivity of the query, thus providing a high number of results. On the contrary GraphDB performs better when specific results are searched, thus when a smaller number of results are requested. For Virtuoso we have found some problems with the *st_intersect* function that is buggy and the queries need to be rewritten using the *st_distance* function. It seems that in this case the optimizer does not use the spatial indexing structure as the starting point for a selection.

ACKNOWLEDGMENT

The authors would like to thanks ONTOTEXT to have provided access to a trial version of their RDF store. This work has been developed in the context of Km4City activity for Sii-Mobility Smart City National project.

REFERENCES

[1] Mulligan, C.E.A.; Olsson, M., "Architectural implications of smart city business models: an evolutionary perspective," *Communications Magazine, IEEE* , vol.51, no.6, pp.80,85, June 2013

[2] P. Bellini, M. Benigni, R. Billero, P. Nesi, N. Rauch, "Km4City Ontology Bulding vs Data Harvesting and Cleaning for Smart-city Services", International Journal of Visual Language and Computing, Elsevier, 2014

[3] Y. Guo, Z. Pan, and J. Heflin. "Lubm: A benchmark for owl knowledge base systems". J. Web Semantics, 3(2-3):158–182, 2005.

[4] Chourabi, H.; Taewoo Nam; Walker, S.; Gil-Garcia, J.R.; Mellouli, S.; Nahon, Karine; Pardo, T.A.; Scholl, Hans Jochen, "Understanding Smart Cities: An Integrative Framework," *System Science (HICSS), 2012 45th Hawaii International Conference on* , vol., no., pp.2289,2297, 4-7 Jan. 2012

[5] C. Bizer, A. Schultz. "The Berlin SPARQL Benchmark". Int. Journal on Semantic Web & Information Systems, Vol. 5, Issue 2, Pages 1-24, 2009

[6] Open Geospatial Consortium, "GeoGeoSPARQL - A Geographic Query Language for RDF Data", Sept. 10 2012 http://www.opengeospatial.org/standards/geosparql

[7] G. Garbis, K. Kyzirakos, M. Koubarakis. "Geographica: A Benchmark for Geospatial RDF Stores". In the 12th International Semantic Web Conference (ISWC 2013). Sydney, Australia, October 21-25, 2013

[8] W3C Consortium, "SPARQL 1.1 Query Language", W3C Recommendation, 21 March 2013, http://www.w3.org/TR/sparql11-query/

[9] O. Erling and I. Mikhailov. "Virtuoso: RDF Support in a Native RDBMS". In Semantic Web Information Management, pages 501-519. Springer, 2009.

[10] G. Ladwig, A. Harth, "CumulusRDF: Linked data management on nested key-value stores", 7th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2011), 2011

[11] K. Kyzirakos, M. Karpathiotakis and M. Koubarakis. "Strabon: A Semantic Geospatial DBMS". In the 11th International Semantic Web Conference (ISWC 2012), Boston, USA, 11-15 November 2012.