An approach to enabling RDF data in querying to invoke REST API for complex calculating

Xianming Zhang
Aviation Industry Development Center of China
forzxm@163.com

Abstract

RDF does not have very good support for calculation, especially complex calculation. SPARQL Inferencing Notation (SPIN) has been proposed with a specific capability of returning a value by executing external JavaScript file that in partly performs complex calculating, however it is still far away from accomplishing many practices. This paper investigates SPIN's capability of executing JavaScript, namely SPINx framework, presents a method of equipping RDF data with a new capability of invoking REST API, by which a user who is querying can obtain returned value by invoking the REST API performing complex calculating ,and then the value is semantically annotated for further use. Calculation of lift coefficient of airfoil is taken as a use case, in which with a given attack angle as input a desired returned value is obtained by invoking a particular REST API while querying the RDF data. Through this use case, it is explicit that RDF data invoking REST API for complex calculating is feasible and profound in both real practice and semantic web.

Keywords: spin, sparql, rdf, rest api

1. Introduction

In past years, a large number of RDF data and RDF-based applications have been developed for various domains. In order to take advantage of semantic feature of RDF, several query and rule languages, such as SPARQL[1], Jena rule (Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A. & Wilkinson, K., 2016) and SWRL (Horrocks, Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. & Dean, M., 2004), have been developed and adopted widely. With these languages users can freely both query and reason about desired information from RDF data, and these languages also provide calculating capability by means of a number of inbuilt functions[2] for users to perform various calculations during either querying or reasoning.

Unfortunately, due to calculation complexity in real world, the calculating capability above is insufficient to accomplish many calculating tasks, the causes are as follows.

- Currently these inbuilt functions fail to perform many complex calculations such as matrix calculation, linear operation, these calculations are essential in such domain as physics and mechanics.

[1] See http://www.w3.org/TR/rdf-sparql-query and https://www.w3.org/Submission/SPARQL-Update - accessed August 23, 2018.

[2] See https://github.com/dotnetrdf/dotnetrdf/wiki/DeveloperGuide-SPARQL-XPath-Functions and http://jena.apache.org/documentation/query/library-function.html - accessed August 23, 2018.

- It is not wise to write too many calculating steps in both query and rule statements, as the feature of the statement is concise and explicit and too many calculating steps often harm this feature.
- Many calculations require external data as source, such as today's temperature for travel decision of today or exchange rate for economic decision, and the existing data either has too large volume to be appended into the new RDF dataset (every high cost and time consumption) or are commercial confidential not to be shared freely by others. Inbuilt functions provided by the languages above only consume data in the related RDF data.

In order to solve such problems this paper turns to SPARQL Inferencing Notation (SPIN)[3]. SPIN is the de-facto industry standard to represent SPARQL in form of RDF, and has been developed out of the necessity to perform calculations on property values. To be pertinent to this paper, SPIN provides a special framework (SPINx) that allows user-defined function to link an external JavaScript file to RDF data by RDF property, performs this user-defined function for calculation by invoking this linked JavaScript file and the resultant value can be semantically annotated by vocabulary of this RDF data. Of course in this situation, the content of linked JavaScript file is simple without many additional functionalities appearing in such working environment as web browser, so that the induced calculating capability is insufficient. But this paper investigates the mechanism of SPINx framework and devises a method to link REST API with RDF data, and invoke REST API while either querying or reasoning. It must be pointed out that this paper opens a profound ground in semantic web technology.

The rest of this paper is as follows: Section 2 presents a use case as motivation to illustrate it is both difficult and useful for a RDF data to deal with complex calculating; Section 3 introduces SPINx framework, especially the working principle of user-defined function linked with an external JavaScript file, and REST API[4]; Section 4 presents a solution to the use case by means the use of a SPINx framework to link with REST API; Section 5 presents the conclusions.

2 Motivation, a Use Case of Calculation for Lift Coefficient of Airfoil

A research group (here called Group A) builds a knowledge base on airfoil, which stores RDF data on airfoil, mainly explicit ones such as airfoil area and shape. An airfoil is designed to provide lift for airplane during flight, so it is necessary for users to query lift coefficient provided by a particular airfoil under a given circumstance. The lift-coefficient formula is as follow:

lift-coefficient=f(attack-angle) (F1)

In F1 there is no explicit formula (or calculation script) to accurately calculate lift coefficient from attack angle. Typically lift coefficient is a list of data through a limited number of experiments that record the data under different attack angles, as shown in Figure 1.

Attack Angle	Lift Coefficient
-5	0
0	0.6
5	1.1
10	1.4
15	1.7
20	1.6
22	0.6
23	0

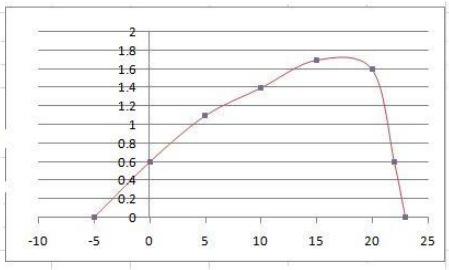

FIG. 1. The table shows a list of attack angles and corresponding lift coefficient after a series of experiments, the graphic presents the curve graph build using the information of the table

[3] See https://www.topquadrant.com/technology/sparql-rules-spin. – accessed August 23, 2018

[4] See https://en.wikipedia.org/wiki/Web_API - accessed August 23, 2018

These experiments are conducted by another research group (here called Group B). The experimental data is locally stored in a web server controlled by Group B and is not freely accessible to others. As a result, the use and maintenance of this knowledge base faces the following problems:

- At present, a user can obtain either a list of data or a curve graph from this knowledge base, and fails to accurately know the lift coefficient of attack angles not shown in the data list, for example attack angles are 2,6,8 etc.
- The experiment is repeated for many times, new experimental data is appended and much of the existing data is modified. Though there is a lasting task of upgrading the knowledge base. The communication cost between Group A and Group B has to be taken into account.
- Group B does not deliver all of the data to this knowledge base for keeping others from getting the comprehensive sense of the experiment. In contrast, Group B allows others to query only one value of lift coefficient each time.

There is a solution to this problem, Group B can develop and deploy a REST API on the web server accessible to others. This REST API implements numerical approximation, such as least square and interpolating to cater for the first one problem, and retrieves data locally stored in web server to cater for the rest of problems. By invoking the REST API, users can obtain the desired results with attack angles and make use of the result for further reasoning. Now devising a unique method to link this knowledge base with REST API and to invoke it in querying (using SPARQL) is the prominent task.

3. Introduction to SPINx, the framework of SPIN for executing JavaScript

3.1 Brief Instruction to SPIN and SPINx

SPIN (SPARQL Inferencing Notation) queries are stored in RDF format and together with RDF data, which makes it possible to share SPARQL queries and update operations with other RDF data. A basic idea of SPIN is to link ontology classes in RDF with SPARQL queries that define constraints and rules formalizing the expected behavior of class members (instances). SPIN has become the de-facto industry standard to represent SPARQL rules and constraints on Semantic Web models and provides meta-modeling capabilities that allow users to define their own SPARQL functions, namely user-defined functions[5].

These user-defined SPIN function are very powerful ways of extending SPARQL, but they are still limited by whatever features are natively supported by the executing SPARQL engine. The SPINx framework included in SPIN makes it possible to define new SPARQL functions that are backed by JavaScript code. Whenever such new functions are invoked, a SPINx-aware SPARQL engine can look up the function's body and execute it using a JavaScript interpreter[6].

3.2 Calculation of the Use Case by means of SPINx Framework

Figure 2 illustrates the working principle of SPINx framework using the concrete use-case of the calculation of lift coefficient. The steps of FIG 2 are described in order as follows.

- Submitting SPARQL and SPINx framework loading RDF data

5 See https://www.topquadrant.com/technology/sparql-rules-spin - accessed August 23, 2018

6 See http://spinrdf.org/spinx.html – accessed August 23, 2018

A user submits a SPARQL statement into SPINx framework, the statement's goal is to calculate lift coefficient with Airfoil: CacuLiftCoefficient being the user-defined function (an instance of spin: Function) and 9 being the input attack angle. After reception the framework starts to search Airfoil: CacuLiftCoefficient in RDF data, and finds a desired segment that is compatible with the statement. Table 1 presents this situation.

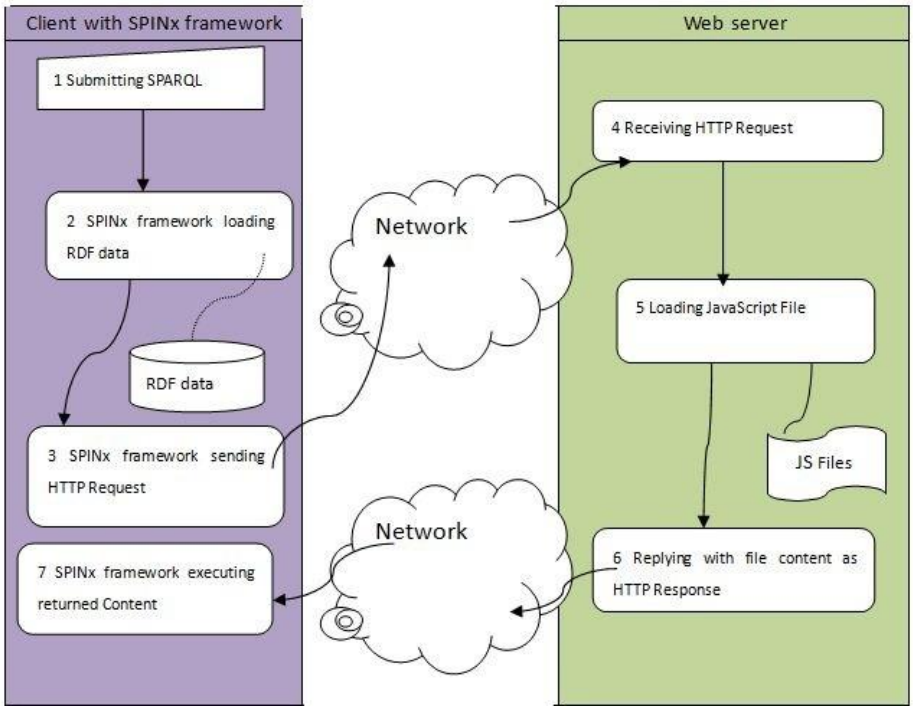

Fig. 1. Working principle of SPINx framework for the use case

TABLE 1: Mapping between RDF data and SPARQL statement

Segment of RDF Data	**SPARQL Statement**

Airfoil:CacuLiftCoefficient
a spin:Function ;
rdfs:subClassOf spin:Function ;
spin:constraint
[rdf:type spl:Argument ;
rdfs:comment "angle attack";
spl:predicate sp:arg1 ;
spl:valueType rdfs:Literal
] ;
spin:returnType xsd:float;
spinx:javaScriptFile "http://web-server/js/calculation.js"
	SELECT ?VALUE
	WHERE {
	BIND(Airfoil:CacuLiftCoefficient(9) AS ?VALUE).
	}

The spin: function Airfoil: CacuLiftCoefficient in the left column is mapped into Airfoil: CacuLiftCoefficient in the right, spl: Argument is mapped into 9, spin: returnType being mapped into ?VALUE means the returned data type is xsd: float, and the URL http://web-server/js/calculation.js is the implementation file of this function.

- SPINx framework sending and web server receiving HTTP request Submitting

After extraction of the URL of JavaScript file, namely http://web-server/js/calculation.js, the SPINx framework automatically sends a HTTP request to a web server identified in the URL. If the web server is connected to the sender/client via network by means of HTTP, it will smoothly receive this request. The excerpt of HTTP request is shown below.

```
[Request-Line]  GET /js/calculation.js
Host          web-server
Accept        text/html,*
Connection     keep-alive
```

- Loading JavaScript file and replying with its Content as HTTP Response

After analyzing the request correctly, the web server automatically retrieves its own file system for the JavaScript file calculation.js, reads content of this file and replies with it as HTTP response to the client.

```
HTTP/1.1 200 OK
Content-Type: application/x-javascript


function CacuLiftCoefficient(arg)
{
// for the sake of brevity, the code is omitted.
return result;
}
```

- SPINx framework executing returned content

After reception of HTTP response, SPINx framework extracts out the function body, semantically reorganize it with the segment of RDF data as well as the submitted SPARQL statement, and finally produces a piece of JavaScript code shown as following that in turn returns resultant value to user after being executed by SPINx framework.

```
return CacuLiftCoefficient(9);
```

3.3 Defect of the use of SPINx framework in the use case

It is impractical to include dataset in the files, especially the volume is not small and not allowed to expose freely.

In order to develop and deploy such JavaScript file, experimental data of confidential will have to be included in this exposed file and be updated frequently for future experiments, which both violates law of information security and is too expensive to maintain the JavaScript file.
As a result, most of applications of SPIN are focused on check constraints, perform data validation and some simple calculation accomplished by user-defined functions within RDF data (Furber & Hepp (2015); Riain & Mccrae (2012); Callahan & Michel (2012); Homola & Serafini (2012)), and its SPINx framework seemingly fails to play its deserved role in SPIN-related applications.

4. Driving SPINx framework to invoke REST API

Although it is impractical for researchers to develop a JavaScript file accessible on the web, but we can turn to an innovative method that is to develop a REST API that returns a small piece of JavaScript code containing resultant value after running. SPINx framework receives and executes this section of code quickly, namely extraction of resultant value in the code.

4.1 Introduction to REST API for the Use Case

A Web API (Application Programming Interface) is typically a defined set of HTTP request messages along with a definition of the structure of response messages for system-to-system interactions (information exchange programmatically)[7]. In this use case, the implementation of REST API is RESTful since it is popular with more and more web applications that have deployed their own REST APIs. Users can access and invoke REST API by means of http://web-server/REST/CacuLiftCoefficient/{AttackAngle}, where {AttackAngle} can be replaced with any real number, such as 9, 9.6 and so on. The REST API for calculating of lift coefficient, developed in Spring Boot, can be written as below

[7] See https://en.wikipedia.org/wiki/Web_API - accessed August 23, 2018

```
@RestController

@RequestMapping(path="/REST")

public class Calculator {

@RequestMapping(Path="/CacuLiftCoefficient/{AttackAngle}"

,produce=MediaType.TEXT_HTML_VALUE)

@ResponseBody

    public  String    CacuLiftCoefficient(@PathVariable("AttackAngle")
float arg1) {

           String result;

     float value


    // for sake of brevity, details are  omitted

    result="function           CacuLiftCoefficient()           {return
"+(String)value+";}";

    return result;

    }


}
```

After accessing to the REST API by URL, client can obtain a piece of code as HTTP response. For example, with http://web-server/REST/CacuLiftCoefficient/9 what client can obtain as follows:

```
function CacuLiftCoefficient () {return 1.34 ;}
```

4.2 Reconstructing working principle of SPINx framework with REST API

After reconstruction, the working principle is as below and for the sake of brevity, just steps in white box are addressed here.

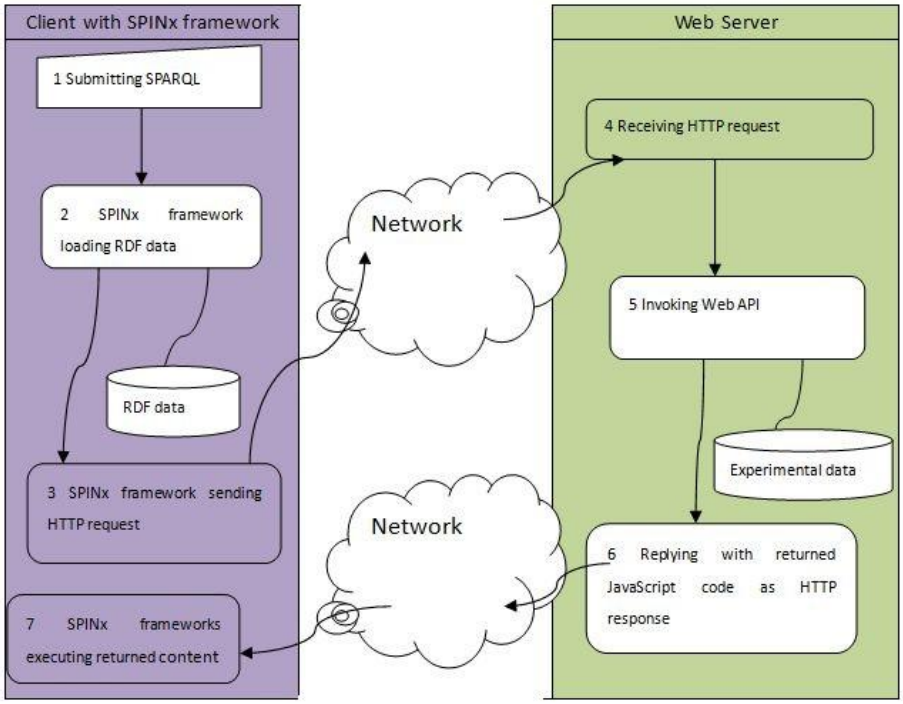

Fig. 1. Reconstructing the working principle of SPINx framework for the use case.

- Submitting SPARQL and SPINx framework loading RDF data

A new spin:Function instance called as Airfoil:CreateURL is added into RDF data, with which URLs for REST API of the use case can dynamically be created with variables of attack angle as input. This user-defined function is shown as below

```
Airfoil:CreateURL
      a spin:Function ;
      rdfs:label "create REST API URL"^^xsd:string ;
      rdfs:subClassOf spin:Functions ;
    spin:constraint
              [ rdf:type spl:Argument ;
                  rdfs:comment "angle attack";
                spl:predicate sp:arg1 ;
                spl:valueType rdfs: Literal
              ] ;
spin:returnType xsd:string;
spinx:javaScriptCode          "return          '          http://web-
server/REST/CacuLiftCoefficient/'+arg1"
```

A user submits SPARQL statement as below before actually beginning to query resultant value for a given input, in which the number 9 is the given input. It is noted that spinx: javaScriptCode links a piece of JavaScript code that be executed locally in querying.

```
DELETE
 { Airfoil:CacuLiftCoefficient  spinx:javaScriptFile ?OLDURL
 }
 INSERT
 { Airfoil:CacuLiftCoefficient  spinx:javaScriptFile ?NEWURL
 }
WHERE{
Airfoil:CacuLiftCoefficient  spinx:javaScriptFile ?OLDURL.
BIND(Airfoil:CreateURL(9) AS ?NEWURL ) }
```

Now the user can query resultant value by submitting SELECT ?value WHERE {BIND(Airfoil:CacuLiftCoefficient( ) AS ?value). }

- Invoking REST API and replying

After analyzing the request correctly, the web server automatically finds and invokes Calculator. CacuLiftCoefficient (9) ,the REST API that organizes the 9 with the experimental data in either data file or database, and processes them with a specific algorithm. After processing, returned value is replied as HTTP response to the client.

```
HTTP/1.1 200 OK
Content-Type: application/x-javascript
function CacuLiftCoefficient(){return 1.34;}
```

It should be noted that such SPARQL operations occur separately, which means the resultant value are independent from each other. Through this example, the feasibility of invoking REST API with SPARQL statements is approved.

5. Conclusions

With abundance in IT infrastructure today, the number of REST API is growing and RDF-based knowledge system should be constructed by fully taking advantage of this situation including REST API rather than from scratch. This paper discusses that usefulness and feasibility of using SPINx framework to invoke REST API for resultant value. It can be said that the paper's achievement is a breakthrough in development of RDF-based knowledge system. In my opinion, the study of this paper can make the semantic web models obtain powerful calculating capacity.

Of course in such situations, the coordinating asynchronous requests, latency, availability and security must be taken into account, these problems should be solved effectively (at least in part) as the technologies for REST API, exemplified by SPRING BOOT, has made much effort to solve them from birth. Many readers familiar with REST API may put forward such viewpoint that majority of REST APIs available are not intended to return a piece of JavaScript code. To solve this problem is to establish a proxy as intermediate between clients and web servers with HTTP as communication protocol. Clients send HTTP request to specific REST APIs on the proxy and the specific REST APIs request normal REST APIs on web servers. In return, the resultant value will be wrapped in a piece of JavaScript code by the proxy and then send back to clients.

Acknowledgements

I would like to thank the reviewers of this article who, with their interest and commitment, contributed to its clarification and improvement.

I would like to thank Mariana Curado Malta and Paul Walk. Since English is not my native language and it is the first time I submit an article to an international conference, I encountered difficulties. They arranged a zoom meeting for me to talk about the conference, Mariana instructed and helped me to typeset this paper, and to reference using the APA style. Paul cleaned my paper for a better understanding. Especially Mariana paid much painstaking care for my paper, and I can not imagine the result without her care. I am truly grateful for their help.

References

Callahan, Alison & Michel Dumontier (2012). Evaluating scientific hypotheses using the SPARQL Inferencing Notation. The Semantic Web: Research and Applications, Lecture Notes in Computer Science 7295, 647-658.

Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A. & Wilkinson, K. (2016). The Jena Semantic Web Platform: Architecture and Design. In HP Laboratories Technical Report HPL-2003-146.

Furber, Christian & Hepp, Martin. (2015). Using SPARQL and SPIN for Data Quality Management on the Semantic Web. In Business Information Systems, Lecture Notes in Business Information Processing, vol. 47, 35-46.

Homola, Martin & Serafini, Luciano. (2012). Contextualized knowledge repositories for the semantic. Journal of Web Semantics , Vol 12, 64-87

Horrocks, Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. & Dean, M. (2004). Swrl: A semantic web rule language combining owl and rule ml. W3C Member Submission, Retrieved August 23, 2018, from http://www.w3.org/Submission/SWRL.

Riain, Sean O & Mccrae, John P. (2012). Using SPIN to Formalize Accounting Regulations on the Semantic Web. In ESWC 2012: The Semantic Web: ESWC 2012 Satellite Events, 58-72.