Semantic data evaluation on federated and distributed systems

Theses of the PhD dissertation
Gergő Gombos
Supervisor: Attila Kiss, Ph.D., C.Sc.

Eötvös Loránd University
Faculty of Informatics
Department of Information Systems
Ph.D. School of Computer Science
Head of school: Prof. Erzsébet Csuhaj-Varjú
Ph.D. Program of Information Systems
Head of Program: Prof. András Benczúr

Budapest, 2018
Introduction

The semantic web [16] is created by Tim Berners-Lee and the aim of it was to store the information on the Internet in a form that can be interconnected with each other. Another aim is the computer can understand the information. Nowadays, the search engines use indexing algorithms which is created by the Google. These algorithms are word-based and the main disadvantage is it does not use the meaning of the word for matching. Since then the Google has also developed its own knowledge base to improve their algorithm. This knowledge base is also based on the expectations of the semantic web.

The W3C drafted the storage structure of the semantic web in the Resource Description Framework (RDF) [18]. Each concept and object must have a unique identifier, that called Internationalized Resource Identifier (IRI). These identifiers are URLs in practice that can be handled by the Internet applications. If the same concept or object are in two different datasets the same IRI has to be used. As soon as we have the IRIs, the information can be written with simple statements. A statement has a subject, a predicate and an object, called RDF triple. We can write also statements with the same subject. In the statements the predicate and the object can be IRIs, and they can be presented in other statements too. Accordingly, the RDF dataset can be presented as a graph. The subject and the object are the nodes, and the predicate is a labeled directed edge between them. We call these graphs Semantic Graph and the stored data is the knowledge base.

The aim of the Linked Open Data (LOD) [22] consortium is to use the information on the Internet as a large knowledge base. To reach this the consortium drafted some requirements for the datasets. The dataset has to be available for everyone and it has to contain identifiers from other datasets to able to connect them. Those datasets are meeting these requirements can be included into the LOD Cloud. The LOD Cloud presents a graph where the nodes are the datasets and the edges are their connections. The size of the nodes is the size of the datasets. The Semantic Data can be accessed through SPARQL Endpoints. The endpoint provides an API to reach the dataset from browsers or applications.

The Semantic Data can be querying with the SPARQL query language, that is similar to the SQL query. It has a SELECT part where we can specify the variables that we want to get back. It has a WHERE part where we specify the conditions for the variables, called triple pattern, that is similar an RDF triple, but some of its attribute are variable. The result will be a subgraph that matches all the conditions. If we want to retrieve information from multiple SPARQL endpoints, the SERVICE keyword can be used. It is a part of the WHERE clause and we have to specify the endpoint that we want to use for
the triple patterns.
Many challenges have been emerged during the implementation of the semantic web. This dissertation focuses on these problems and provides practical solutions. 1.) One of the problems is the accessibility. The semantic web is difficult from the point of the user because they needs to know the URL of the SPARQL Endpoints and they needs to know the structure of the datasets behind the endpoint. The federated systems provide a solution for these problems because they give a centralized accessibility for the datasets. 2.) Another problem it the size of the semantic data. If the users try to reach the dataset with mobile phones or if they want to ask complex queries, then they can do only difficulty. These problems are solved by client-server systems and the Big Data tools.

Theses

1 Semantic data on client-server architecture

Section 3 of the dissertation shows the semantic web usage in mobile environment. The new mobile devices and the mobile Internet give opportunities to use the semantic data on mobile. The first devices do not have strong hardware for large data. Today, they have enough resources for both storing and processing the data, but it is not enough for the million of triples of the semantic web. On the other hand when the device processes the data, the battery life will significantly be reduced. The solution is the client-server architecture, where the costly calculations and data storage are on a strong computer, and the client does only the presentation. The book called 'The semantic web: real-world applications from industry' [20] presents some examples: how we can use the semantic web in the industry. It contains examples for various fields: economic, health, education or even security applications. Most of the applications are implemented server-side and are accessible via web browsers. The first and the most popular client-side application is the DBpedia Mobile [21], which is a geo-based semantic application. This means it uses the location information of the user to offer attractions that is close to the user.

Thesis 1 (A server supports the weak client (mobile) to access the semantic data and to run complex and federated semantic queries. [1, 2, 5]). Because the limit of the mobile devices (memory, CPU, data traffic, battery) they can not reach the semantic data. The client-server architecture is useful for this problem. Where the server manage, ask or join the semantic data and task of the client is only the visualization.

The dissertation presents three applications that based on the thesis. The first ap-
plication (Section 3.3) is a viewer for the industry data, where the data were stored in semantic form [1]. The second one is an indoor navigation application, where the objects were stored in semantic form [2] (Section 3.4), and finally the semantic browser for mobile environment [5] (Section 3.5).

2 Semantic data in federated environment

The Section 4 presents the federated systems. As I mentioned in the introduction, the main problem with semantic web is that the user needs to know the URL of the SPARQL endpoints and they needs to know the structure of the datasets. Understanding of the data is difficult because it is unstructured and it has large size. The federated systems can help in these problems because with these we do not need to know the endpoint or the structure of the dataset. The queries that are using more than one endpoints are called federated queries. Rakhmawati et al. [30] presented the architecture of the federated systems. In the federated system we can write SPARQL query that does not have SERVICE keyword in the WHERE part. The endpoint selection is based on the triple patterns. There are two techniques for this selection. The first one is based on asking and query all the endpoints and they response true if it has a triple or false it does not have (ASK technique). The second one stores information about the endpoints and it uses these information for the decision (document technique). This technique stores usually the predicates of the endpoints. For example the DARQ [19] (distributed ARQ) uses the ASK technique and the FedX [25] uses the document technique. In Section 4.3 I describe the general federated systems with the ASM model [26] and I refine them for two specific case: semantic browser and recommendation system. The usage was inspired by the ASM model of the Grid Systems [17], which is similarly distributed and parallel like the federated systems.

Another problem with the semantic web is the query writing without the knowledge of the datasets. Hoefler [28], Campinas [31] and Han [33] wrote that the usage of the semantic web is difficult, because we can not write SPARQL query easily. In Section 4.5 I present a solution, which helps to the user to write SPARQL queries.

Thesis 2 (Method for triple patterns recommendation with federated systems that helps for the user. [4] The formal description of the federated systems and the this recommendation created with the ASM model. [4]) The federated systems can collect triples and create triple patterns. The recommendation based on the triple patterns of the query that the user wrote. The system recommends predicates based
on the type of a variable. These recommendations can help to the user. The ASM model describes formally our requirements for a system. The main steps of the federated systems can be formulated with the ASM. The main part of a federated system: query parser, endpoint selection, run of the subqueries and production of the result.

The document technique typically uses the predicates of each endpoints. In Section 4.4 I present a solution where I use the namespace of the endpoints for the endpoint selection.

Both of the two endpoint selection techniques have problem. The ASK technique runs on every endpoint with every triple pattern. The document technique has an information file that we need to update every time. If we use the recommendation system we will get information about the endpoints and the datasets. This thesis is detailed in Section 4.6.

Thesis 3 (The recommendation system gives information that the endpoint selection of the federated system can use. [7] The namespaces of the semantic data can be used for endpoint selection. [3]). The endpoint selection is based on some information about the endpoints. It is typically the predicates of the datasets. The datasets usually have a unique namespace (prefix), which is the prefix of the IRIs. These information also can be used for the endpoint selection. On the other hand the recommended triple patterns have predicate and namespace informations. The endpoint selection part of the federated system can use these information. It does not need to check all the endpoints with every triple patterns.

3 Connection between the semantic data and the distributed systems

The Section 5 focuses on with the problem of the semantic web based on their size. The first result is presented in Section 5.3, that is related to the unknown data structure problem. The visualization tools can help for this issue. There are some related works for semantic data viewer: LODmilla [32] or VizBoard [27]. A solution is when we present the data with a graph editor and we can understand the semantic graph. The problem is that these tools can not handle large graphs. The technique that is presented in the dissertation uses the bisimulation [29] to reduce the graph size. The bisimulation merges the similar nodes. I implemented our method with the MapReduce algorithm, which is designed for large data processing.
Thesis 4 (With bisimulation we can reduce the graph size. The graph visualization tools can be handle this smaller graph. The bisimulation technique can be parallel with the MapReduce algorithm. [6].). We will can understand the data, if we use a graph visualization tool. This tool can not handle these data because of their size. A solution for this problem is a bisimulation. It merges the similar nodes to one node. This algorithm works only on small graphs, so we need to extend it to large graphs. We used the Big Data tools and the MapReduce algorithm for large graphs.

The connection between the semantic data and the Big Data tools is a new research area. The aim of that is to store the large knowledge base in the Big Data tools and querying the data from these systems. More related works are presented in the SPARQL evaluation on Big Data tools (Hadoop MapReduce [23], Pig [24]). These systems can work effectively in a cluster and the algorithms is similar to the divide-et-impera. The problem with these tools is the disk usage because they use to much disk read and write operations, and the disk is a slow hardware. The Spark is a new tools, that implemented a new storage technique. It uses the memory of the cluster to store the object during the data processing, which is more effective. Another advantage of the Spark is it has more libraries for stream processing, machine learning or graph processing. Since, the semantic data is a graph, I created an algorithm for SPARQL evaluation that works on the Spark GraphX (graph processing tool). In Section 5.4 I presented the algorithm and how to store the semantic data in a distributed graph processing system. I presented an evaluation plan that will help for the Spark GraphX. The idea behind the distributed graph processing is the message passing between the nodes on the edges. The query plan gives the direction of the messages in an iteration. I compared my results with the S2X [34] that created by Schätzle et al.

Thesis 5 (SPARQL evaluation on Spark GraphX with linear query plan (Spark(q))l [8]). The Spark GraphX distributed graph processing tool is usable for storing and querying the semantic data. The linear query plan works in one triple pattern in one iteration. The only way to work the algorithm we have to give the right order of the edges. The algorithm can be faster if we decrease the necessary messages. We can do this a simple check before the message sending. We send message only if the sender node has the subresult of the previous iterations. Other problem is the handling of the Data property. When we analyze the triple patterns we do not know which are Data property. We check it during the evaluation with a loop edge. The nodes send message to itself. The GraphX works on the active edges in every iteration. One edge is active if one of its nodes got message in the previous iteration. For this we have to keep alive the edges until
they would we processed. We do this with dummy messages, called ‘alive’.

In Section 5.5 I presented the improved version of the algorithm, that uses parallel query plan for the evaluation.

Thesis 6 (SPARQL evaluation on Spark GraphX with parallel query plan (P-Spark(q)l) [9]). The SPARQL evaluation on Spark GraphX can be faster if we work more than one triple pattern in one iteration. If we do this, then we do not need the dummy messages to keep alive the edges, because we use all the edges from a level of the query tree, that is a tree representation of the query plan. The problem is with this technique is the message handling, because now the messages have different schemes. We need to union some messages and to join others. We can skip the loop edge in this algorithm, because we can decide which predicate is Data property during the graph reading. We can collect more statistics about the graph that help us to improve the algorithm. We calculate the number of the predicates and we use it for the query plan. We use that predicate first, which is the least in the dataset.
Related publications of the author


Other publications of the author


References


[33] Lushan Han et al. “Querying RDF data with text annotated graphs”. In: Proceedings of the 27th International Conference on Scientific and Statistical Database Management. ACM. 2015, p. 27.

[34] Alexander Schätzle et al. “S2X: Graph-Parallel Querying of RDF with GraphX”. In: Proc. of 1st International Workshop on Big-Graphs Online Querying. 2015.