Stage de Master 2 – 2017

Linked Open Data Query Processing

Designing a SPARQL query engine on top of a scalable cluster computing platform

Encadrants : B. Amann, H. Naacke

Contact : bernd.amann(@)lip6.fr

Durée : 6 mois à partir de février 2017

Contexte et Motivation

The Semantic web community aims to digitalize the human knowledge and thus promotes standards for managing semantic information (RDF, RDF/S, *etc.*). This dissemination effort has the positive effect of increasing international awareness of the vital role of the Linked Open Data cloud (LOD) and of its potential for raising many societal challenges. To turn this awareness into positive action, radical improvements making the LOD a universal accessible source of knowledge are required that will integrate LOD queries into a wide range of decision frameworks. Although LOD browsing services already exist (*cf.* Google Knowledge Graph), they only allow for basic single predicate search, and do not cover the entire LOD, thus deliver incomplete answers. There is a need for a more complete and expressive LOD query engine, that should be efficient as well.

Ce stage se base sur les récents travaux effectués au laboratoire LIP6 et présentés à la conférence BDA 2016. Une nouvelle solution pour évaluer efficacement des requêtes SPARQL sur des milliards de triplets a été conçue et mise en œuvre sur un cluster comprenant une vingtaine de machines. Les premiers résultats montrent le bénéfice de la solution et ouvre de nouvelles perspectives qui font l'objet de ce stage

Objectifs

Les requêtes SPARQL servant à explorer les bases de connaissance du Linked Open Data ont, pour la plupart, des motifs particulièrement étoffés : les requêtes contiennent des chemins longs et ramifiés. Par exemple, une requête pour connaitre le profil sémantique d'une personne (entourage, activités favorites, etc...) est un motif avec plusieurs dizaines de variables de jointures. Une telle requête s'avère très longue à calculer dans un environnement distribué à cause des nombreux transferts de données qui en découlent.

L'objectif principal du stage est de proposer une solution pour limiter au maximum les transferts de données pendant l'évaluation des requêtes complexes. Deux aspects sont étudiés:

- **Stockage et organisation des données RDF**. Comment organiser (*i.e.,* fragmenter, répliquer et placer) les données pour réduire la durée d'exécution d'une requête ? Le compromis à étudier est le suivant : (i) d'une part tenir compte de certaines caractéristiques typiques des requêtes et des données pour organiser les données de manière à favoriser les calculs locaux sans transfert, (ii) d'autre part réduire le temps nécessaire à la préparation des données préalablement à leur interrogation, en s'appuyant sur une organisation générique des données indépendante des requêtes et des données. Une piste possible est d'exploiter les calculs effectués lors des requêtes précédentes pour améliorer graduellement le placement (et le degré de réplication) des données fréquemment interrogées.

- **Evaluation des requêtes SPARQL**. Le traitement efficace des requêtes SPARQL nécessite de choisir et combiner astucieusement plusieurs algorithmes de jointure. Après avoir recensé (à travers un état de l'art) les algorithmes de jointure parallèle les plus récents et estimé leur coût en termes de transferts de données, il s'agit de proposer des stratégies pour construire le plan d'exécution d'une requête SPARQL de telle sorte que le coût soit réduit et que le plan puisse s'exécuter directement sur une plateforme de calcul existante (Spark ou Flink). En particulier, plusieurs modes d'exécution différents pourront être considérés: produire le résultat complet d'une requête, calculer seulement les N éléments du résultat (N étant de l'ordre de la dizaine), évaluer des requêtes de type ask, select ou construct. Selon le mode, déterminer le ou les algorithmes de jointure apportant le plus d'efficacicté.

Plan de travail

Le candidat pourra s'il le souhaite se concentrer sur l'un ou l'autre des deux objectifs présentés ci-dessus. Le travail à effectuer se décompose en plusieurs phases :

- Etat de l'art (à partir des références bibliographiques ci-dessous) sur l'évaluation de requêtes Sparql dans les RDF stores distribués.
- Proposer une solution pour analyser les requêtes les plus fréquentes. En considérant que les données sont distribuées sur plusieurs machines, proposer une stratégie de réplication des triplets qui améliore la localité des traitements pour évaluer plus rapidement les requêtes.
- Proposer une solution pour déterminer un plan d'exécution d'une requête SPARQL dont le coût est quasi minimal.
- Implantation et expérimentation avec les données de DBPedia et les traces des requêtes du service d'accès [3], en utilisant la plateforme Apache Spark ou Apache Flink et le cluster de calcul du laboratoire.

Bibliographie

[1] O. Curé, G. Blin. RDF Database Systems: Triples Storage and SPARQL Query Processing, 2015

[2] O. Curé, H. Naacke, T. Randriamalala, B.Amann. LiteMat: a scalable, cost-efficient inference encoding scheme for large RDF graphs. Biggraph Workshop at BigData Conference 2015.

[3] O.Curé, H.Naacke, B.Amann. SPARQL Query Processing with Apache Spark, BDA 2016

[4] DBPedia France Sparql endpoint. http://fr.dbpedia.org/sparql et http://dbpedia-test-fr.inria.fr/dbpedia-sparql

[5] Montoya, G., Skaf-Molli, H., Molli, P., & Vidal, M. E. (2015, October). Federated SPARQL Queries Processing with Replicated Fragments. In *The Semantic Web-ISWC 2015-14th International Semantic Web Conference*.

[6] Alexander Schätzle, Martin Przyjaciel-Zablocki, Simon Skilevic, Georg Lausen: S2RDF: RDF Querying with SPARQL on Spark. PVLDB 9(10): 804-815 (2016)