PDSM: Pregel-Based Distributed Subgraph Matching on Large Scale RDF Graphs

Qiang Xu[†§], Xin Wang[†§*], Yueqi Xin[†§], Zhiyong Feng[‡§], Renhai Chen[†§]

[†]School of Computer Science and Technology, Tianjin University, Tianjin, China
[‡]School of Computer Software, Tianjin University, Tianjin, China
[§]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
{xuqiang3,wangx,xinyueqi,zyfeng,renhai.chen}@tju.edu.cn

ABSTRACT

This paper presents a novel Pregel-based Distributed Subgraph Matching method PDSM to answer subgraph matching queries on big RDF graphs. In our method, the query graph is transformed to a spanning tree based on the breadth-first search (BFS). Two optimization techniques are proposed to filter out part of the unpromising intermediate results and postpone the Cartesian product operations in the Pregel iterative computation. The extensive experiments on both synthetic and real-world datasets show that PDSM outperforms the state-of-the-art methods by an order of magnitude.

1 INTRODUCTION

RDF can be represented as a labeled, directed graph consisting of a set of triples (s, p, o), where s is the *subject*, p the *predicate*, and o the *object*. Given an RDF graph G and a query graph G_Q, subgraph matching is to find subgraphs over G that satisfy all the triple patterns in G_Q, which is a *conjunctive query* (CQ). It is known that the complexity of CQ is NP-complete. For instance, the CQ $Q_1(?x) \leftarrow (?x, \texttt{rdf:type}, \texttt{Student}) \wedge (?x, \texttt{takesCourse}, ?y) \wedge (?x, \texttt{advisor}, ?z) \wedge (?z, \texttt{teacherOf}, ?y)$ over the RDF graph G_1 in Figure 1 finds those students who take courses that their advisors teach, and one of the query results is highlighted in green.

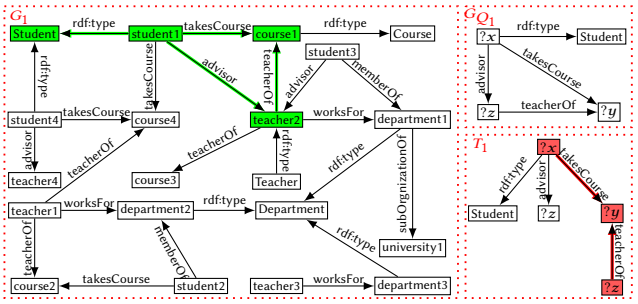

Figure 1: An example RDF graph G_1 and a CQ Q_1 over G_1

* Xin Wang is the corresponding author.

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3186907

There has been some research works on subgraph matching in distributed environments. In [1, 3], a query graph is decomposed into a set of subqueries, i.e., triple patterns and STwigs; partial results of these subqueries are joined together to form the final answer. However, the performance of these methods highly depends on decomposition and join order of the subqueries, thus a large number of joins may incur expensive costs. Although PSgl [2] employs graph traversal without joins, it cannot be easily adapted to the characteristics of RDF graphs.

We propose a join-less method PDSM which transforms a query graph to a spanning-tree variant MHT (Minimum Height Tree), which is then matched against an RDF graph *in parallel* using Pregel, a *vertex-centric* parallel graph computational model. The experiments have verified the efficiency and scalability of our method.

2 PROBLEM AND OUR ALGORITHM

Let V, E, and Σ denote the set of vertices, edges, and edge labels in an RDF graph G, respectively. Assume there exists an infinite set Var of variables disjoint from V, and the name of every element in Var starts with the character ?, e.g., $?x \in Var$. A CQ Q over G is defined as: $Q(z_1, ..., z_n) \leftarrow \bigwedge_{1 \le i \le m} tp_i$, where $tp_i = (x_i, a_i, y_i)$ is a *triple pattern*, $x_i, y_i \in V \cup Var$, $a_i \in \Sigma$, $z_j \in \{x_i \mid 1 \le i \le m\} \cup \{y_i \mid 1 \le i \le m\}$. A CQ Q is also referred to as a query graph G_Q, e.g., G_{Q_1} in Figure 1. Let $\bar{x} = (x_1, ..., x_m)$, $\bar{y} = (y_1, ..., y_m)$, and $\bar{z} = (z_1, ..., z_n)$. The semantics of CQ Q over G is: (1) μ is a mapping from vertices in $\bar{x}$ and $\bar{y}$ to vertices in V; (2) $(G, \mu) \models Q$ iff $(\mu(x_i), a_i, \mu(y_i)) \in G$ and x_i and y_i match $\mu(x_i)$ and $\mu(y_i)$, respectively; and (3) Ω_Q is the answer set of Q over G, i.e., the set of $\mu(\bar{z})$, such that $(G, \mu) \models Q$.

We first transform a query graph G_Q to an MHT T, which is a tree rooted at v_r, the *central vertex* in the *diameter* of G_Q, to minimize the height of T. The breadth-first search is used to construct T from v_r. First, children of v_r are added to T, i.e., all triple patterns (v_r, p_i, v_i) or $(v_i, p_i, v_r) \in G_Q$ are copied to T. Let $Child(v)$ denote the set of children of node v and $v_i \in Child(v_r)$. Next, for each node v_i, children of v_i are generated in T similarly by exploring the edges that are not visited yet in G_Q. This process continues until all edges in G_Q are visited. For example, T_1 is the MHT of G_{Q_1} in Figure 1. Note that there exists duplicate nodes in T if G_Q contains a circle, e.g., node $?z$ in the MHT T_1. For each node u in T, let $\Omega(u)$ be the partial matching results of the subtree rooted at u, thus $\Omega(v_r)$ is the answer set of G_Q.

Our join-less method employs the Pregel model to match nodes of an MHT T against G in a bottom-up manner. Given an RDF graph G, each vertex $v \in G$ is in either *active* or *inactive* state, and the

function voteToHalt makes v to deactivate itself. The Pregel *parallel computation* is defined as a sequence of iterations, i.e., *supersteps* separated by global synchronization barriers, on active vertices. In our PDSM algorithm, (1) in superstep 0, all vertices are active, every vertex v matches leaves of T *in parallel* and sends the partial matching results of these leaves to their neighboring vertices (lines 7-9); (2) in the following supersteps, vertices who receive massages Msg are reactivated and match parents of those nodes matched in the previous superstep (lines 10-17). Similarly, these vertices send messages of partial matching results of subtrees rooted at these parents (line 17); and (3) this process is iterated until the root is matched (line 16). We can see that one level of the tree T is matched in each superstep. Moreover, $\mathtt{val}(v) = (M_P, Res)$ denotes the value of vertex v, where M_P is the set of $\Omega(u)$ and Res is the set of query results. We have proved that PDSM is correct and can finish in d supersteps, where d is the height of T.

Algorithm 1: PDSM

```
   Input  : An RDF graph G, the MHT T of a CQ Q
   Output : The answer set Ω_Q
1  Ω_Q ← ∅, Msg ← ∅;
2  for superstep = 0 to d do                    // for each v ∈ V
3      every active vertex v calls compute(v, Msg) in parallel;
4  foreach v ∈ V do Ω_Q ← Ω_Q ∪ val(v).Res ;
5  return Ω_Q;
6  Function compute (v, Msg)
       // superstep 0, match all leaves
7      foreach leaf l in T ∧ l matches v do
8          add a pair (l, v) to an empty mapping μ;
9          sendMessage(v, l, {μ});          // {μ}, i.e., Ω(l)
       // superstep 1 to d, handle received messages
10     foreach Ω(u) ∈ Msg do
11         if the parent u' of node u matches v then
12             add a pair (u', v) to an empty mapping μ;
13             M_p ← M_p ∪ {Ω(u)};          // v receives Ω(u)
14             if v receives all Ω(u_i) and u_i ∈ Child(u') then
15                 merge μ and mappings in Ω(u_i) to get Ω(u');
16                 if u' = v_r then Res ← Res ∪ Ω(u') ;
17                 else sendMessage(v, u', Ω(u')) ;
18     return voteToHalt(); // deactivate v at the end
19 Function sendMessage (v, u, m)          // m is a message
20     foreach neighboring vertex v' of v do
21         if (u, parent(u)) is the same as (v, v') then
22             sMsg(v', m);          // send a message m to v'
```

As to the optimization strategies, we employ the W3C *RDF Shapes* standard as the *priori* knowledge to filter out part of the unpromising intermediate results. Furthermore, we also decompose an MHT into a set of paths to postpone Cartesian product operations. For example, one such path of the MHT T_1 in Figure 1 is highlighted in red. Since the matching of these paths can be executed *in parallel*, the query efficiency is independent of a specific matching order.

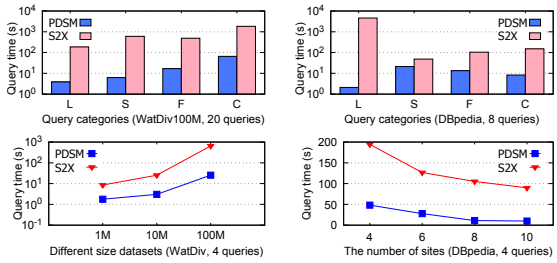

Figure 2: Comparison of PDSM and S2X

3 EXPERIMENTS

Our prototype system, implemented in Scala using Spark GraphX, is deployed on a 10-site cluster. Each site has one CPU with 4 cores, 16GB memory, and 250GB disk storage. We use Hadoop 2.7.4 and Spark 2.2.0. All the experiments were carried out on Linux (64-bit CentOS) operating systems. For the datasets, we used the WatDiv benchmark synthetic data and the DBpedia real-world data (23.4M triples). RDF queries are grouped into four categories according to their shapes, including *linear*, *star*, *snowflake*, and *complex*. In WatDiv, the benchmark contains 20 basic query templates. Due to the absence of query templates in DBpedia, we designed 8 queries, covering all four query categories mentioned above.

Figure 2 illustrates the efficiency and scalability of PDSM by comparing with S2X. We can observe that PDSM demonstrates a much better query efficiency on both datasets and all query categories. The main reason is that our join-less PDSM, based on Pregel, employs graph traversal to avoid the expensive cost of join operations. Furthermore, we randomly selected one query from each of the four categories to verify the scalability of PDSM. Varying the size of WatDiv from 1M to 100M, query times of both methods have increased, however, the growth rate of query times of S2X is higher than that of PDSM. With the number of sites varying from 4 to 10, the speedup ratio of PDSM is about 1.71 times of S2X.

4 CONCLUSION

In this paper, we proposed the join-less method PDSM for efficiently answering the subgraph matching problem on large-scale RDF graph data using Pregel. Our extensive experimental results on both synthetic and real-word datasets have verified the efficiency and scalability of our method, which outperforms the state-of-the-art subgraph matching method S2X by one order of magnitude.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (61572353, 61672377) and the Natural Science Foundation of Tianjin (17JCYBJC15400).

REFERENCES

[1] Alexander Schätzle, Martin Przyjaciel-Zablocki, Thorsten Berberich, and Georg Lausen. 2015. S2X: graph-parallel querying of RDF with GraphX. In *VLDB Workshop on Big Graphs Online Querying*. Springer, 155–168.
[2] Yingxia Shao, Bin Cui, Lei Chen, Lin Ma, Junjie Yao, and Ning Xu. 2014. Parallel subgraph listing in a large-scale graph. In *Proc. SIGMOD*. ACM, 625–636.
[3] Zhao Sun, Hongzhi Wang, Haixun Wang, Bin Shao, and Jianzhong Li. 2012. Efficient subgraph matching on billion node graphs. *Proceedings of the VLDB Endowment* 5, 9 (2012), 788–799.