SparkRDF: Elastic Discreted RDF Graph Processing Engine With Distributed Memory

Xi Chen, Huajun Chen, Ningyu Zhang, and Songyang Zhang

College of Computer Science, Zhejiang University,
Hangzhou 310027, China
`{xichen,huajunsir,zxlzr,syzhang1991}@zju.edu.cn`

Abstract. With the explosive growth of semantic data on the Web over the past years, many large-scale RDF knowledge bases with billions of facts are generating. This poses significant challenges for the storage and retrieval of big RDF graphs. In this paper, we introduce the SparkRDF, an elastic discreted semantic graph processing engine with distributed memory. To reduce the high I/O and communication costs for distributed platforms, SparkRDF implements SPARQL query based on Spark, a novel in-memory distributed computing framework. All the intermediate results are cached in the distributed memory to accelerate the process of iterative join. To reduce the search space and memory overhead, SparkRDF splits the RDF graph into the multi-layer subgraphs based on the relations and classes. For SPARQL query optimization, SparkRDF generates an optimal execution plan for join queries, leading to effective reduction on the size of intermediate results, the number of joins and the cost of communication. Our extensive evaluation demonstrates the efficiency of our system.

Keywords: Big RDF Graph, SPARQL, SPARK, Distributed memory.

1 Introduction

With the development of Semantic technologies and Web 3.0, the amount of Semantic Web data represented by the Resource Description Framework (RDF) is increasing rapidly. Traditional RDF systems are mainly facing two challenges. i)scalability: the ability to process the big RDF data. Most existing RDF systems are based on single node[4][1], which are easily vulnerable to the growth of the data size because they usually need to load large indexes into the limited memory. ii) real-time: the capacity to implement SPARQL query over big RDF graph in near real time. For highly iterative SPARQL query, existing MapReduce-based RDF systems suffer from high I/O cost because of iteratively reading and writing large intermediate results in disk[3].

In this paper, we introduce SparkRDF, an elastic discreted RDF graph processing system with distributed memory. It is based on Spark, a in-memory cluster computing system which is quite suitable for large-scale real-time iterative computing jobs[5]. SparkRDF splits the big RDF graph into MESGs(Multi-layer Elastic SubGraph) based on relations and classes by creating 5 kinds of indexes(C,R,CR,RC,CRC) with different grains to cater for diverse triple patterns(TP). These index files on demand are modeled as RDSG(Resilient Discreted SubGraph), a collection of in-memory semantic subgraph objects partitioned across machines, which can implement SPARQL query by a series of basic operators. All intermediate results(IR), which are also regarded as the RDSG, remain in the distributed memory to support further fast joins. Based on the query model, several corresponding optimization tactics are then presented.

The remaining of this paper is organized as follows. Section 2 introduces the index data model and iterative query model of *SparkRDF*. In Section 3, we present the results of our experiments. Finally, we conclude and discuss the future work in Section 4.

2 SparkRDF

2.1 Index Data Model: MESG

We create the index model called MESG based on relations and classes, which extends traditional vertical partitioning solution by connecting class indexes with predicate indexes, whose goal is to construct a smaller index file for every TP in the SPARQL query. At the same time, as it is uncertain that the class information about the entities can are given in the SPARQL query, the SparkRDF needs a multi-layer elastic index scheme to meet the query need for different kinds of TP. Specifically, we first construct the class indexes(C) and relation indexes(R). Then a set of finer-grained index files(CR,RC,CRC) are created by joining the two kinds of index files. All the index files are stored in the HDFS.

2.2 RDSG-based Iterative Query Model

For SparkRDF, all the index files and IRs can be modeled as an unified concept called RDSG(Resilient Discreted SubGraph). It is a distributed memory abstraction that lets us perform in-memory query computations on large clusters by providing the following basic operators: RDSG_Gen, RDSG_Filter, RDSG_Prepartition, RDSG_Join. Figure 1 illustrates the RDSG-based query process. Every job corresponds to one query variable.

2.3 Optimization techniques

Based on the data model and query model, several optimization strategies are made to improve query efficiency. First, TR-SPARQL refers to a Type-Restrictive SPARQL by passing variable's implicit class message to corresponding TPs that contains the variable. It cuts down the number of task (remove the TPs whose predicate is rdf:type)and the cost of parsing every TP(form a more restrictive index file). Then we use a selectivity-based greedy algorithm to design a optimal execution order of TPs, greatly reducing the size of IR. At last, the location-free prepartitioning is implemented to avoid the shuffling cost in the distributed join. It ignores the partitioning information of index files, while repartitioning the data with the same join key to the same node.

3 Evaluation

We implement the experiment on a cluster with three machines. Each node has 16 GB DDR3 RAM, 8-core Intel Xeon(R) E5606 CPUs at 2.13GHz. We compare SparkRDF with the state-of-the-art centralized RDF-3X and distributed HadoopRDF. We run the RDF-3X on one of the nodes. HadoopRDF and S-parkRDF were executed in the cluster. We use the widely-used LUBM dataset with the scale of 10000, 20000 and 30000 universities, consisting of 1.3 , 2.7 and 4.1 billion triples. For the LUBM queries, we chose 7 representative queries which are roughly classified into 2 categories: highly selective queries (Q4,Q5,Q6) and unselective queries(Q1,Q2,Q3,Q7). A short description on the chosen queries is provided in the Appendix.

Table 1 summarizes our comparison with HadoopRDF and RDF-3X(best times are boldfaced). The first observation is that SparkRDF performs much better than HadoopRDF for all queries. This can be mainly attributed to the following three characteristics of SparkRDF: finer granularity of index scheme, optimal query order and effective memory-based joining. Another observation is that SparkRDF outperformed RDF-3X in Q1,Q2,Q3,Q7, while RDF-3X did better in Q4,Q5,Q6. The result conforms to our initial conjecture: RDF-3X can achieve high performance for queries with high selectivity and bound objects or subjects, while SparkRDF did well for queries with unbound objects or subjects, low selectivity or large intermediate results joins. Another result is that RDF-3X fails to answer Q1 and Q3 when the data set size is 4.1 billion triples. On the contrary, SparkRDF scales linearly and smoothly when the scale of the datasets increases from 1.3 to 4.1 billion triples. It proves that SparkRDF has a good scalability.

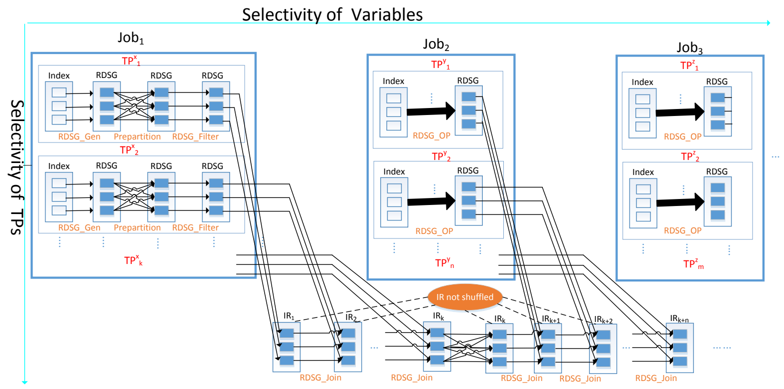

Fig. 1. The Iterative Query Model of SparkRDF

Table 1. Performance Comparison in seconds for SparkRDF(SRDF), HadoopRD-F(HRDF) and RDF3X.

	LUBM-10000			LUBM-20000			LUBM-30000		
	cluster systems		centralized system	cluster systems		centralized system	cluster systems		centralized system
	SRDF	HRDF	RDF3X	SRDF	HRDF	RDF3X	SRDF	HRDF	RDF3X
Q1	**478.5**	8475.4	2131.4	**1123.2**	>3h	4380.3	**1435.4**	>3.5h	failed
Q2	**11.9**	3425.2	13.8	**25.8**	>2h	28.9	**40.3**	>2.5h	43.5
Q3	**1.4**	6869.7	24.6	**1.4**	>2.5h	90.7	**1.4**	>3h	failed
Q4	14.4	11940.3	**0.7**	23.8	>4h	**0.8**	32.5	>8h	**0.8**
Q5	6.8	2587.5	**0.7**	10.9	>1h	**0.7**	13.0	>3h	**0.7**
Q6	10.3	7210.5	**0.6**	16.4	>2.5h	**0.7**	20.0	>3h	**0.7**
Q7	**54.6**	1911.2	101.5	**112.5**	>0.7h	198.5	**201.3**	>1h	853.0

4 Conclusion and Future Work

In the paper, we introduce the SparkRDF, a real-time scalable big RDF graph processing engine. Also We give some the experimental results to show effectiveness of the SparkRDF. In the future, we would like to extend the work in few directions. First, we will handle more complex SPARQL patterns(such as OPTIONAL). Finally, we will make a more complete and comprehensive experiment to validate the efficiency of SparkRDF.

References

1. Atre, M., Chaoji, V., Zaki, M.J., Hendler, J.A.: Matrix Bit loaded: a scalable lightweight join query processor for rdf data. In: Proceedings of the 19th international conference on World wide web. pp. 41–50. ACM (2010)
2. Guo, Y., Pan, Z.: LUBM: A benchmark for owl knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web 3(2), 158–182 (2005)
3. Husain, M., McGlothlin, J., Masud, M.M., Khan, L., Thuraisingham, B.: Heuristics-based query processing for large rdf graphs using cloud computing. Knowledge and Data Engineering, IEEE Transactions on 23(9), 1312–1327 (2011)
4. Neumann, T., Weikum, G.: The rdf-3x engine for scalable management of rdf data. The VLDB Journal 19(1), 91–113 (2010)
5. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. pp. 2–2 (2012)

APPENDIX

We provide the SPARQL queries used in the experimental section:

Q1-Q6 are the same as [1]. Q7 corresponds to the Q14 of [2].