A Distributed Partitioning Algorithm for E-Government Service Platform based on Cloud Computing

Yonglin Leng[1,a], and Xiaohong Sun[2,b]

[1]College of Information Science and Technology, Bohai University, Jinzhou 121000, China

[2]College of Mangement, Bohai University, Jinzhou 121000, China

[a]lengyonglin@qq.com; [b]sxh@qq.com

Keywords: E-government; Network public opinion; RDF; Partitioning

Abstract: Network public opinion from the perspective of E-government Crisis management can effectively govern the virtual society. The monitoring and prediction of public opinion can not be separated from the data. The single-machine model is inefficient for the large number of public opinion data. The emergence of distributed technology based on cloud computing improves the processing efficiency of public opinion. But a key technology of distributed processing is the data partitioning. In order to mine the public opinion data more deeply, RDF (Resource Description Framework) is used to describe network resource, which can achieve more efficient and accurate retrieval. This paper will study how to partition the network public opinion data based on RDF. The star fabric existing extensively is an important structure of RDF data. Aiming at the star fabric, we propose a distributed partitioning algorithm based on star-based. First, the 2 hops star fabric is built with Hadoop. And next a weighted graph is constructed based all these star fabrics, and a balance K-medoids clustering algorithm is adopted to divided the weighted graph. Experimental result gets a lower replication ratio and a better load balancing.

1. Introduction

Improving the management level of virtual society is a necessary requirement for governments at all levels to strengthen and innovate social management and enhance the ability of social governance [1]. The influence of network public opinion on the real society is gradually strengthened. At the same time, the outbreak of the crisis of network public opinion has posed a severe challenge to the ability of government governance. Network public opinion from the perspective of E-government Crisis management can effectively enhance the government's ability to govern the virtual society and it has great practical significance to build a harmonious public opinion environment on and off the internet [2]. The monitoring and prediction of public opinion can not be separated from the data. A large number of public opinion data can not be processed effectively by the traditional single-machine model. The emergence of distributed technology based on cloud computing has solved the problem of data processing efficiency of public opinion. One of the important technical problems for distributed processing is the partition of big public opinion data. Because the big public opinion data comes from the network. In order to mine the public opinion data more deeply, semantic web technology uses resource description framework (RDF) to express network resource, and achieves seamless links of human, machine and web resources by establishing connections between smaller data which can achieve more efficient and accurate retrieval. This paper will study how to partition the network public opinion data based on RDF.

The essence of RDF graph makes different partitioning schemes which would influence the retrieval efficiency of RDF data. A large number of queries analyzed by Gallego et al. certifies that 60% joins types are star fabric [3]. This star fabric reflects the linkages between entity and their attributes. So it is very important to the data partitioning and retrieval of Semantic Web of Things. This paper treats the star fabric as partitioning granularity and creates a weighted graph based on

  DOI: 10.25236/meici.2019.106

shared nodes. Then a *k*-way balanced partitioning algorithm is used to divide the weighted graph. Finally, the star fabrics are mapped to each partition. The experiment evaluates the weighted graph partitioning algorithm from both horizontal and vertical perspectives. The results show the weighted graph partitioning is superior in replication ratio partition efficiency.

2. Related Work

Graph partitioning algorithm is to distribute the vertices with high correlation to the same storage node and reduce the communication between storage nodes in data query [4, 5]. The traditional Kernighan Lin (KL) algorithm uses iterative dichotomy to realize k-way partitioning. FM algorithm improves the efficiency of KL by moving a node instead of switching nodes. In addition, some researchers combine simulated annealing [6], genetic algorithm [7] and other typical algorithms to improve the performance of partitioning. But these algorithms still have efficiency problems for large-scale RDF graph.

Multilevel partitioning algorithms, known for theirs high efficiency and quality of partitioning, have been widely used in RDF graph, such as Metis [8]. Huang et al. use Label Propagation algorithm (LPA) to rough the vertices of graphs, and used METIS to achieve the final partition [9]. BRGP [10] uses modularization as label update rule, and uses the energy attenuation strategy of label to balance of subgraph. However, all these algorithms only consider the node information of the graph, and regard the node or edge as the partition granularity.

Some algorithms are based on the idea of subgraph structure. Lee et al. [11] proposed a general scalable Graph Partitioning Framework SPA, which created vertex blocks for each vertex in a graph by pressing out, in or two-way edges. Then the vertex block is extended in the form of one or n hops. Finally, the vertex blocks are partitioned by hash or minimal cut. Although SPA considers how to avoid duplication when partitioning vertex blocks, the partition results still produce a large number of duplicates, especially for the partitioning of edge-entry vertices, which will seriously affect the balance of segmentation and the rate of partitioning duplicates.

3. The Weighted Graph Algorithms

In RDF graph, most vertices both includes the outgoing and incoming edge. The analysis results on LUBM, SP2Bench and DBLP datasets show that the outgoing edges have a more even distribution than the incoming edges. Such as LUBM datasets, the incoming in some vertices are even more than one million, while the maximum outgoing edges are only 18. If the star fabric is built on the incoming edges, there will a large number of uneven distribution of star fabric and replication. In addition, the SPARQL query statistics also shows that only a few O-O connections are found in the query, so this paper chooses star fabric based on outgoing edges to construct the partition granularity and extends it to two hop to improve the query speed. At the same time, in order to reduce the replication data generated during partition, a weighted graph based on vertices and edges is created, and the distributed repository of star fabric is realized by using the BELP balanced partitioning algorithm mentioned in [10].

3.1 Constructing Partitioning Unit.

In an RDF graph, the star fabric is represented as $S_v=\{V_s, E_s, L_s\}$, where V_s is the set of vertices and E_s is the set of edges in S_v. The central point is v and V_s-v are the leaf nodes. The central point and the leaf node satisfy $V_s = \{v\} \cup \{v_i | v_i \in V_s < v, v_i > \in E_s, 1 \le i \le |E_s|\}$. L_s is the label set of E_s.

If the SPARQL is a star fabric, each storage node can execute the SPARQL query independently and concurrently. After that, the master node completes the merging of query results without any communication among storage nodes. However, a complex query can not be implemented directly in parallel. The master node need to perform multi-round star fabric distribution, so there are a large number of communications between the master node and the storage node. A n-hops star fabric is proposed by Lee et al. [11], which can execute a *n*–hops SPARQL query concurrently. But t the larger

n is, the replication ratio will be higher. Furthermore, statistics show that most queries can be decomposed into 1 to 2-hops by one round. Therefore, this paper extends the basic star fabric by 2 hops, which gets an acceptable query performance between the replication size and the communication.

Hadoop MapReduce is used to construct the basic and extended star fabric. Hadoop need to execute two rounds Map and Reduce functions. In the first round HDFS will distribute a fixed-size data block to each map function. And the map function returns the key-value pairs (*s*, (*po*)) and (*o*, (*ps*)). In the key-value pair, 0 and 1 are added to distinguish (*s*, (*po*)) or (*o*, (*ps*)). Then, the reduce phase merges all the key-value pairs with the same key to form a list of key values:

$$S^1 = (key_{so}, ((p_1^1, o_1^1, 1), \dots, (p_m^1, o_m^1, 1), (p_1^0, s_1^0, 0), \dots, (p_n^0, s_n^0, 0))) \quad (1)$$

Next, each map function of second round receives the list of key values S^1 coming from last round. Then S^1 is reassigned to get another key-value pair by the flag 0 or 1. For example, the flag 0 in S^1 is converted to the key list:

$$S_i^2 = (s_i^0, (p_i^0, key_{so}((p_1^1, o_1^1), \dots, (p_m^1, o_m^1))), \ i = 1, 2, \dots m \quad (2)$$

For the flag 1, we can remove it directly. The expression is formatted as follows:

$$S_i^2 = (key\_so, ((p_1^1, o_1^1), \dots, (p_m^1, o_m^1))) \quad (3)$$

Finally, the reduce function merges again all key-value pairs with the same key into a key-value list:

$$S^2=(key, ((p_1^1, o_1^1, 1), \dots, (p_m^1, o_m^1, 1), (p_i^0, key\_so((p_1^1, o_1^1), \dots, (p_m^1, o_m^1))) \quad (4)$$

3.2 Distribution of Star Fabric.

Through the star fabric, we get the basic granularity unit of RDF. And then, these basic granularity units would be distributed to different computing nodes. There are two rules for the distribution. First, it should satisfy the load balancing, because the load balancing may impact the query efficiency. Second, the replications of vertex. The granularity units include some common vertices, if these common vertices lie in the same storage node, the cost of storage would decrease dramatically and the efficiency of retrieval would be improved.

To satisfy the two rules, a weighted graph is constructed based on these basic granularities, and a clustering algorithm is employed to achieve the weighted graph partitioning.

Let $G_w = (V_w, E_w)$ be a weighted graph, where $V_w=\{S_1, S_2,\dots, S_n\}$ is the set of weighted vertices coming from the basic or extended 2-hops star fabric. The weighted vertex is denoted as:

$$w(S_i) = \sum_{u \in S_i} w(u) \quad (5)$$

If shared vertices exist between two weighted vertices, a weighted edge is added. The weighted edge is described as:

$$w(S_i, S_j) = \frac{|SS_i \cap SS_j|}{|SS_i \cup SS_j|} \quad (6)$$

The weighted of vertex represents the scale of star fabric, and the weighted of edge represents the similarity between two weighted vertices.

Next, we use a balanced K-medoids clustering algorithm in reference to partition the weighted graph.

4. Experiments

In this section, three representative datasets are used to evaluate the weighted graph partitioning algorithm. Table 1 gives the relevant information of datasets. The datasets of LUBM and SP^{2}Bench adopt benchmark generators to get the benchmark RDF datasets and Uniprot is a real dataset, which is a protein dataset.

The experiment environment includes a master node and 16 storage nodes. The master node has 20GB memory and a 2.00GHZ $\times 24$ Inter Xeon processor. Each storage node has the same

configuration with a 4GB memory, 2.4GHZ Inter Xeon processor and 500GB disk. The construction of star fabric is executed on a Hadoop system, which runs on the platform of JDK 1.6.0 version. The partitioning is executed on the master node.

Table 1 Basic Information of Datasets

Dataset	#Vertex	#Edge	#Predicate
LUBM2000	66,059,204	276,345,040	18
SP^2B-100M	42,125,032	113,246,098	93
Uniprot	139,942,781	687,025,165	84

We selected two comparison algorithms: Minimum cut and Hash. Minimum cut is implemented by METIS, and maps and splits 2-hops fabric on the master node. Hash partitioning is directly executed on the master node.

Partitioning Performance. Fig.1 shows the partitioning efficiency of three partitioning algorithms. Obviously, Minimum cut has the lowest partitioning efficiency. The reason lies in that the partition of Minimum cut using METIS starts from the original graph. And then, it realizes the mapping between the star fabric and the partition results. As described in Fig.1, with the increasing of RDF graph scale, the most obvious change is Minimum cut among three algorithms. The efficiency of Hash partitioning is the fastest, but it has a large of replications. Because Hash partitioning ignores the correlation of the vertices. The weighted graph not only considers the association between vertices, but also restricts the replication scale by 2-hops star fabric. So the partitioning efficiency are improved obviously.

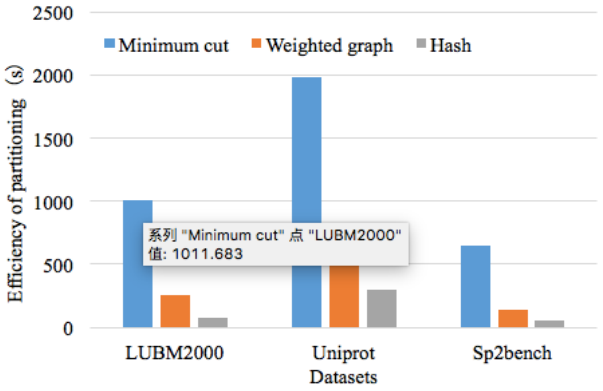

Figure 1. Partitioning efficiency

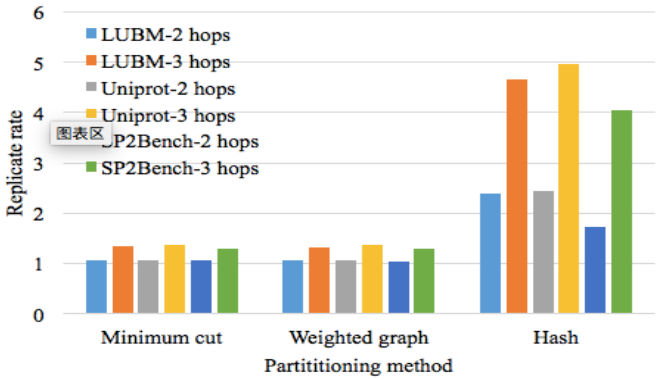

Figure 2. Replication Rate

The replication rate about the triples is calculated on three datasets with 2 hops and 3 hops. Fig.2 shows the result. The Hash method has more than 2 times replication rate than Minimum cut and Weighted graph partitioning methods on 2 hops. Moreover, it can be seen that the 3 hops in Hash method has even higher replication rate. So Minimum cut and Weighted graph partitioning methods are effective in cutting down triple replicas. Though, there is little gap in Minimum cut and Weighted graph partitioning methods, but the partitioning efficiency and balance indicate that the weighted is superior to minimum cut. Meanwhile, the larger the hop is, the higher the triple replica will be.

Fig.3 describes the triple distribution of storage node by using three different partitioning methods. The 2 hops star fabric is constructed on the outing edge. As can be seen that the Hash partitioning method has the best balance, but the fluctuation of minimum cut is maximum. The main reason is the order of the division and the vertices mapping. Although Metis framework is good at the partitioning balance, but the inhomogeneous phenomenon still makes a great difference. The weighted graph partitioning method is better than the minimum cut, but it also lower than Hash partitioning, which also proves that the effectiveness of hash partitioning on the balance partitioning.

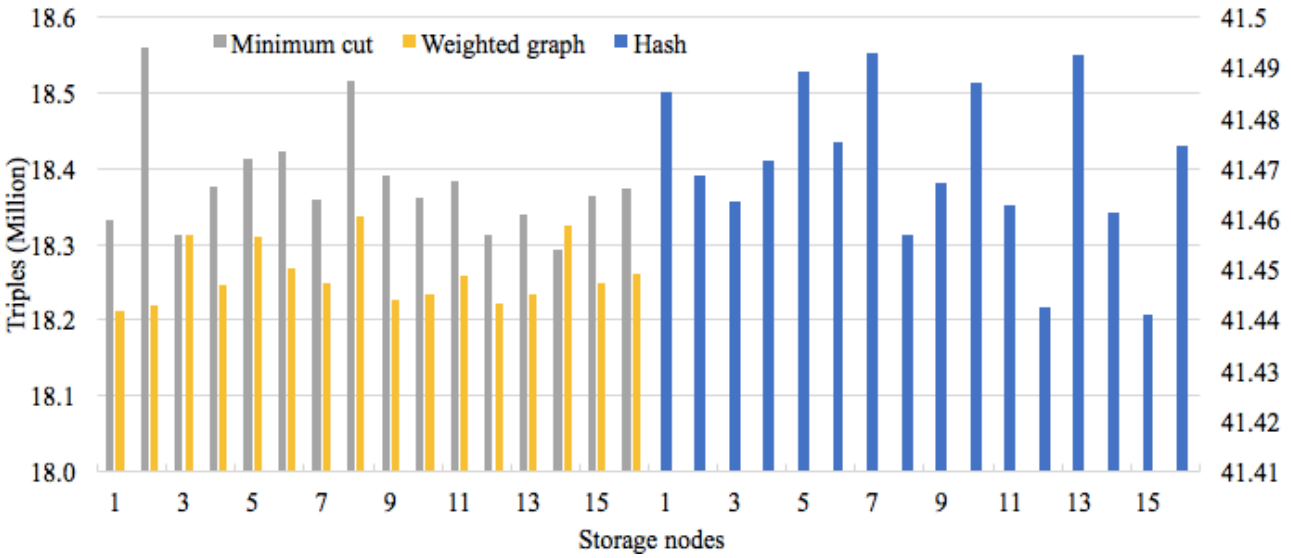

Figure 3. The balance on computing nodes

5. Conclusion

The monitoring and prediction of public opinion can not be separated from data. In recent years, there is a heterogeneous trend for public opinion data. RDF is an effective data model for describing heterogeneous data. This paper mainly studies the problem of data partitioning in the distributed storage of large-scale public opinion data. Star fabric is a common sub-structure of RDF data, which effectively reflects the relationship between data. In this paper, star fabric is used as RDF data partitioning granularity, and RDF data partitioning algorithm is studied. The experimental results show that the overall performance of the weight graph partitioning algorithm based on star fabric is better than that of the minimum cut and hash methods in the replication rate and load balancing.

Acknowledgements

This work is partly supported by Liaoning Social Science Fund (No. L14AGL002, L13AGL002) and the Science and Technology Project of the Liaoning Provincial Education Department (No. LQ2017004).

References

[1] H. Deng, K. Karunasena, X. Wei: I*nternet Research*, Vol. 28(2017) No.4, p.11.

[2] W. Zhou, J. Han, Y. Gao: *Concurrency & Computation Practice & Experience*, Vol. 28(2016) No. 3, p. 729.

[3] K. Lee, L. Liu: *Proceedings of the Vldb Endowment*, Vol.6(2013) No. 14, p. 1894.

[4] T. Bosch T, B. Mathiak: *International Journal of Metadata, Semantics and Ontologies*, Vol.8 (2013) No. 3, p. 254.

[5] J. Kim J, H. Shin, H. Chafi: *Proceedings of the Vldb Endowment*, Vol. 8(2015), No. 11, p. 1238.

[6] M.A. Ahmed, T.M. Alkhamis: *Computers & Operations Research*, Vol. 29(2002), No. 4, p. 387.

[7] X. Hu, T. H. Chu, H. C. Chan: *IEEE Transactions on Emerging Topics in Computing*, vol. 1(2013), no. 1, p. 148.

[8] G. Karypis, V. Kumar: *International Cryogenics Monograph*, (1998), p. 121.

[9] L. Wang, Y. Xiao, B. Shao: *Proc, IEEE 30th International Conference on Data Engineering*,

(Chicago, USA, March 31-April 4, 2014). p. 568–579.

[10] Y. Leng, Z. Chen, F. Zhong: *Concurrency & Computation Practice & Experience*, vol. 29(2017), no. 14, p.11.

[11] K. Lee, L. Liu, Y. Tang: *Proc, IEEE Sixth International Conference on Cloud Computing* (Santa Clara, CA, USA, June 28-July 3, 2013), p. 327.