Removing the Redundancy from Distributed Semantic Web Data

Ahmad Ali Iqbal[1,2], Maximilian Ott[2], and Aruna Seneviratne[1,2]

[1] School of EE&T, University of New South Wales (UNSW), Australia
[2] National Information and Communication Technology Australia (NICTA)

Abstract. Peer-to-peer databases have proven to be an effective way for sharing data. However, distributed knowledge management in P2P databases brings a variety of non-trivial challenges along with its benefits. Such challenges include determining the right content provider(s) and removing the duplicate data transfer if a relatively larger portion of data is redundant and is made available in distributed providers. The aim of this paper is to address data redundancy removal problem such that excessive bandwidth usage due to in-network duplicate data transfer can be minimized. We provide analytical and experimental evaluation of our schemes in terms of the number and size of the packets that flow in the network while keeping confidence level of results high.

1 Introduction

As the tremendous growth of peer-to-peer networks for file sharing continues to strain, research community continues to focus on its use for knowledge sharing [4] in order to exploit its further benefits. P2P databases like Xpeer [7] and AmbientDB [3] emerge as a way for retrieving data from an unstructured and decentralized P2P network and overcome the limitations posed in distributed databases by eliminating constraints like static topology and heavy administration requirements. Such databases have diverse information stores to serve the individual clients and thus, these stores are often not well integrated with each other and they collect the data in uncooperative environment. Due to having no control over information stores, duplicate data storage across multiple stores becomes a serious problem for the clients who search through these stores and costs for redundant data transfer.

These systems demand the following two core operations (i) discovery of information stores for a given query also known as *resource selection* and (ii) redundancy removal in order to minimize bandwidth usage while transferring the data between peers. Resource selection algorithm determines the most suitable information stores for a query provided as an input and gives the ranked list of the candidates. Resource selection [4][5][8] has been under the focus in database community since last decade and is out of scope of this paper. The aim of this paper is to study data redundancy removal problem when it is made available from distributed Resource Description Framework (RDF) based information stores. In order to achieve this, we present summary exchange algorithms

P. García Bringas et al. (Eds.): DEXA 2010, Part I, LNCS 6261, pp. 512–519, 2010.

for coordination between peers that employ bloom filter. They are widely used in variety of database and networking applications [1] to minimize the size of message exchange at the cost of compromising confidence level. Due to the space limitation, we omit the operation of bloom filter and refer [1][2] to the interested readers.

Next, we define *confidence level* (CL) as accuracy of results from distributed stores. Formally, it is defined as a function of $1 - FP_{avg}$ where FP_{avg} is the average of false positive results from the distributed information systems for a given query. We define *average response time* (ART) as the delay between the query generation and final result preparation by the information receiver after merging the results received from distributed information stores. *Selection Set* (SS) is a special type of message used by information receiver to inform the information stores about the selection of element that receiver expects to receive.

Rest of this paper is structured as follows; Redundancy removal schemes are presented in section 2. In section 3, we compare the performance of these schemes. Finally, we conclude this paper in section 4.

2 Duplication Removal

In order to reduce the duplicate data transmission on the network, we present 3 flavors of summary exchange algorithms and discuss their improvement in comparison with brute-force approach. The prerequisite of these algorithms is a *resource selection algorithm* [5] that gives the ranked list of potential information stores.

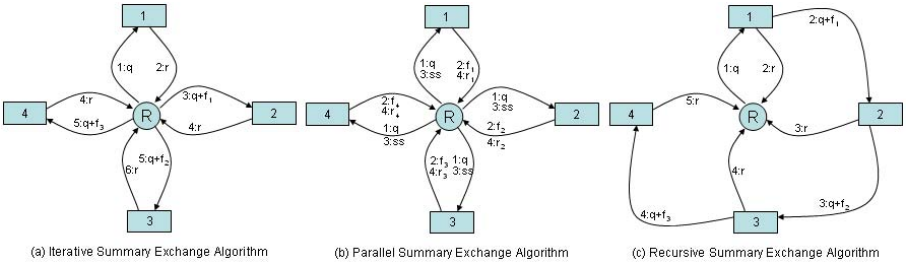

Fig. 1. Flow of Messages in ISE, PSE and RSE algorithms

2.1 Iterative Summary Exchange (ISE) Algorithm

This is an evolutionary algorithm where information receiver (R) monotonically receives the set of RDF statements from multiple information stores. In this algorithm, information receiver iteratively approaches the information stores and gets part of relevant results to a query as follows;

1. Information receiver sends the query (q) to the information store that was ranked highest in the list of information stores for q.

2. Information store executes the query onto its repository and send the matching RDF statements (r) to information receiver.
3. Information receiver creates m bit bloom filter on the results it has received for query q using k hash functions such that k hashes are created on each statement of the result set and relevant bits are marked in the bloom filter.
4. Information receiver sends the query, bloom filter ($q + f_n$ refers to both) and configuration parameters (CP) to information store ranked next in the ranking list for a query q. CP defines all those parameters which are used to create the bloom filter like size, number and type of hash functions.
5. Information store executes the query to retrieve the results.
6. Despite transmitting all matching RDF statements like highest ranked information store did, this store filters out the results indicated by the bloom filter and sends only the set of those statements which were not marked in the received bloom filter. For testing the membership of statements, information store uses CP received from information receiver (see step 4).
7. Step 3 to 6 repeat for the rest of the information stores within the list of potential candidates for a given q.
8. Finally, information receiver combines all the received RDF statements in order to prepare final result for q.

The message flow between information receiver and distributed stores is shown in figure 1a. Suppose that there are l information stores that contain the data matching a query q and n_i is the number of RDF statements received by R after iteration i, each iteration is a combination of querying and receiving the results from exactly one information store and the number of bloom filters created by ISE algorithm is $l - 1$. Then confidence level for a given query q when ISE algorithm is used can be calculated by

$$CL_q := 1 - \frac{\sum_{i=0}^{l-1} \left[1 - \left(1 - \frac{1}{m}\right)^{k \cdot n_i}\right]^k}{l - 1}. \tag{1}$$

2.2 Parallel Summary Exchange (PSE) Algorithm

In PSE algorithms, individual information stores evolve independently of their repositories from their peers thus query execution for creating and membership testing of RDF statements is straightforward and thus decreases the waiting time. In this algorithm, receiver's query is executed in parallel on selected information stores as follows;

1. Information receiver multicast the query to all information stores selected by a resource selection algorithm.
2. Those stores create bloom filter on the matching RDF statements to the receivers query.
3. Information stores send those filters to the receiver.
4. Information receiver distinguishes between corresponding set bits of the all received bloom filters for unique RDF statements and duplicate RDF statements by performing *AND* operation and chooses the candidate information store for each duplicate RDF statement.

5. Selection of RDF statements that each information store will provide is informed by information receiver.
6. Each information store provides the RDF statements according to the selection notified by information receiver

Explanation. PSE algorithm needs more message passing and processing from bloom filter as compared to ISE algorithm as shown in figure 1b. It is because the receiver has to decide the information store(s) for overlapping RDF statements based upon their corresponding bloom filters. In order to inform the selection of contents, IR transmits SS message to the ISs where $SS \leq bloomfiltersize$.

Suppose that there are l information stores that contain RDF statements matching a query q and unlike in equation 1, n_i is the number of RDF statements matching a query q in ith information store and total bloom filters created in PSE algorithm are l. Then CL_q for a q can be calculated by

$$CL_q := 1 - \frac{\sum_{i=0}^{l-1} \left[1 - \left(1 - \frac{1}{m}\right)^{k \cdot n_i}\right]^k + C_i}{l}. \tag{2}$$

where C_i is the constant boosting the false positive probability due to multiple hashing algorithms used to create the bloom filter. The ground truth about the bloom filter is that it uses more than one hashing algorithms and they are not perfect thus generate false positives. The problem with PSE when more than one hashing algorithms are used is that the bit set by one hashing algorithm for one element in one IS may overlap with the corresponding bit of another hashing algorithm for another element in another IS. This problem is known as *bit conflict* that further supplements false positive probability. In order to deal with this issue, we partitioned the total bloom filter space into k subspaces such that each hashing algorithm uses only its assigned subspace. This way the scope for each hashing algorithm in a bloom filter becomes independent from others.

2.3 Recursive Summary Exchange (RSE) Algorithm

The basic idea of RSE algorithm is that ISs gossip (negotiate) among themselves in order to coordinate and reduce the redundant data transfer cost. Flow of messages between information stores and receiver is shown in figure 1c. The algorithm works as follows;

1. Information receivers unicast a query (q) along with configuration parameters (CP) to IS that has the highest priority for a given query q. As mentioned earlier priority list of IS for a specific query q is the output of *resource selection algorithm* that runs in advance for each query. CP contains the size of bloom filter (m), number (k) and selection of hashing algorithms.
2. The information store with the highest priority executes the query q and retrieve the matching RDF statements from its repository.
3. Information store creates the bloom filter on the retrieved RDF statements using CP.

4. Information store removes itself from the priority list and sends those statements to the receiver and bloom filter along with CP to the next information store in the priority list.
5. Despite sending all RDF statements as the highest ranked information stores do, this store filters out the results indicated by the bloom filter and send only the set of those statements which were not marked by the information receiver. For testing the membership of statements with the bloom filter, information store uses CP received from information receiver (see step 1).
6. This information store updates the previously received bloom filter by adding the entries for RDF statements retrieved from its local repository. For this sake, it uses the CP received from information receiver.
7. Step 4 to 6 repeat until all the IS are removed from the priority list.

In order to formalize confidence level of the system, assume that there are l information stores that contain RDF statements matching a query q and like equation 2, n_i is the number of RDF statements matching a query q in ith information store then CL_q can be calculated by

$$CL_q := 1 - \frac{\sum_{i=0}^{l-1} \left[1 - \left(1 - \frac{1}{m}\right)^{k \cdot n_i}\right]^k}{l-1}. \tag{3}$$

2.4 Discussion

These algorithms have their own pros and cons. ISE algorithm is more simplistic while it is suitable for applications where ART is a trivial factor. On the other hand, in PSE and RSE algorithms, participating information stores process queries faster thus reduces latency for end users. In PSE algorithm, participatory information stores process the query in parallel and receiver decides the portion of information it receives from but in RSE algorithm, information stores negotiate with each other thus they have more control on the information that they provide than receiver. In view point of false positive probability, RSE algorithm outperforms ISE and PSE algorithms because each RDF statement distributed across l information stores for a query q is used only once for creating a bloom filter (see equation 3) and there is no additional probability. In contrast, PSE algorithm generates higher false positive probability in comparison to ISE and RSE because of the additional false positive (see equation 2).

3 Evaluation

Although, there is no other published work on distributed ontology based redundancy removal to be compared with ours, packet and message level redundancy removal for pub/sub system has been discussed for years. We use the following performance metrics: ART, message exchanges, data transfer cost and confidence level of results. This experiment is setup on native simulator [1] developed in Java

[1] `http://ee.unsw.edu.au/~z3197878/simulator/`

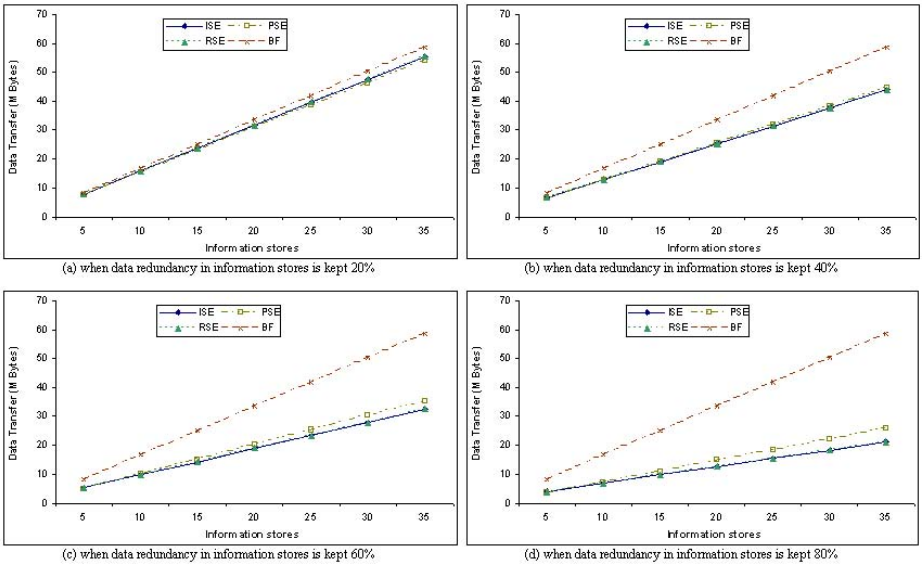

Fig. 2. Data transfer in distributed information retrieval when varying the data redundancy in ISs

and the simulation is developed using a grid topology of 20x20=400 nodes. We transfer n times 104858 elements of size 128 bits each where n is the number of ISs that varies from 5 to 35 with the increment of 5. We used 9 hash functions (i.e. k=9) using [6] for the bloom filter and kept bloom filter size (m) $2 * n * k$. The reason for choosing this m is to achieve upto 99% confidence level of results as stated in [2]. We kept the size of SS message equal to m for the sake of simplicity, however, it can be compressed further for further reduction in overhead traffic. Furthermore, we set 20% data redundancy between IS contents where it is not explicitly stated otherwise.

Figure 2 compares our redundancy removal algorithms in comparison with brute-force approach in the presence of 20, 40, 60 and 80 % data redundancy in ISs. In order to answer receiver's query, total data transfer by varying the number of ISs is shown in figure 2. It is clear that after a certain threshold percentage (reasonably small value) in data redundancy, summary exchange algorithms outperforms over brute-force and once this threshold is achieved, improvement in terms of data transfer becomes proportional to the percentage of redundant data in distributed ISs as shown in figure 3(a). Please note that our experiments are based on 20% data redundancy except the ones shown in figure 2 and 3(a).

ART for ISE as shown in figure 3(b), is more that other algorithms because IR queries ISs one after the other and waits for the response from one before sending it to another. In contrast, PSE algorithm outperforms others in terms of ART because query in a set of ISs is executed in parallel and receiver simultaneously gets the redundancy removed RDF statements. The reason that PSE gets better response time over brute-force is due to (i) lesser data transfer

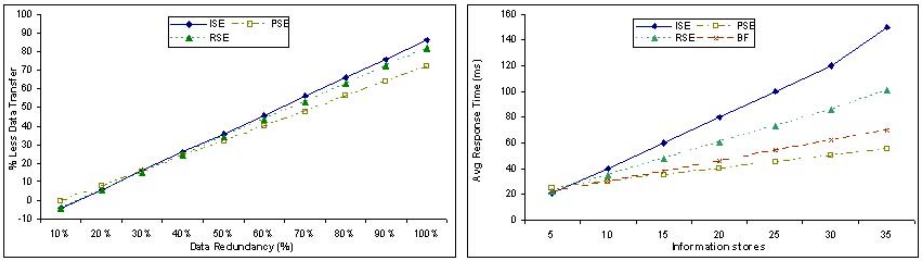

Fig. 3. (a) Summary of improvement in terms of data transfer reduction with redundancy removal algorithms in comparison to brute-force approach, (b) Effect on ART by varying the number of distributed ISs

and (ii) no redundancy removal by the receiver during merging time after receiving from distributed ISs. On the other hand, ART for RSE is slightly less than ISE because ISs in ISE are contacted sequentially by the IR while in RSE ISs directly communicate with each other. This certainly saves waiting time for message exchange between ISs and IR for RSE.

Figure 4 shows the total data transfer trend in order to achieve confidence level from 95% to 100%. False positive in this setting yields in missing the data which should actually be delievered due to matching the receiver's query. Thus, in order to get higher confidence level as depicted in figure 4, larger size of the bloom filters are required to transfer, causing more data transfer in terms of overhead traffic. In case of PSE, overhead traffic exponentially increases because in addition to regular false positive generated by the bloom filter, it is further supplimented during partioning mechanism.

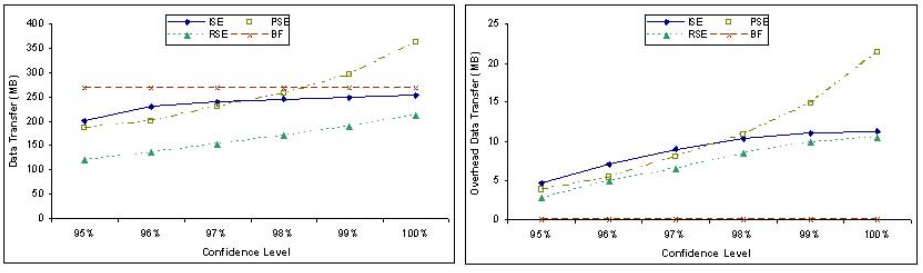

Fig. 4. To achieve higher confidence level, more overhead data have to transfer and it makes significant increase on total data transfer

4 Conclusion

This paper develops and analyzes three redundancy removal algorithms from distributed semantic information stores. In contrast to brute-force, these schemes are not focused on finding the perfect solutions but rather on finding good enough

solutions. The algorithms reduce upto 80% data transfer cost in the presence of duplicate data in distributed information stores. Furthermore, these algorithms have different characteristics and can achieve higher confidence level of results at the cost of overhead traffic. In comparison to brute-force, reduction in data transfer cost in the presence of redundant data in distributed information stores is proportional to redundancy among the data in these stores and if data redundancy crosses the threshold (approx. 15%) percentage, cut in cost for data transfer can be achieved.

Acknowledgments

NICTA is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs.

References

1. Broder, A., Mitzenmacher, M.: Network applications of bloom filter: A survey. Internet Mathematics I(4), 485–509 (2003)
2. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: A scalable wide-area web cache sharing protocol. IEEE Transactions on Networks 8(3) (2000)
3. Fontijn, W., Boncz, P.: Ambientdb: P2p data management middleware for ambient intelligence. In: PERCOMW'04, USA (2004)
4. Haase, P., Siebes, R., Harmelen, F.: Peer selection in peer-to-peer networks with semantic topologies. In: International Conference on Semantics of a Networked World: Semantics for Grid Databases (2004)
5. Iqbal, A., Ott, M., Seneviratne, A.: Resource selection from distributed semantic web stores. In: Int. Conf. on Data and Knowledge Engineering (2010)
6. Kirsch, A., Mitzenmacher, M.: Less hashing, same performance: Building a better bloom filter. In: European Symposium on Algorithms (2006)
7. Sartiani, C., Manghi, P., Ghelli, G., Conforti, G.: Xpeer: A self-organizing xml p2p database system. In: Workshop on P2P and Databases (2004)
8. Si, L., Callan, J.: Relevant document distribution estimation method for resource selection (2003)