Efficient Top-*k* Query Processing Algorithms in Highly Distributed Environments

Qiming Fang[a,b], Guangwen Yang[b]
[a] School of Computer, Hangzhou Dianzi University, Hangzhou 310018, China.
[b] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
Email: fangqiming@gmail.com; ygw@tsinghua.edu.cn

***Abstract*—Efficient top-*k* query processing in highly distributed environments is a valuable but challenging research topic. This paper focuses on the problem over vertically partitioned data and aims to propose more efficient algorithms.. The effort is put on limiting the data transferred and communication round trips among nodes to reduce the communication cost of the query processing. Two novel algorithms, BulkDBPA and 4RUT, are proposed. BulkDBPA is derived from the centralized algorithm BPA2 which requires very low data access. BulkDBPA borrows the idea of best position from BPA2 and so has the advantage of low data transferred. It further reduces the communication round trips by utilizing bulk read and bulk transfer mechanism. 4RUT is inspired by the algorithm TPUT which only requires three communication round trips to get the exact top-*k* results. 4RUT improves its top-*k* lower bound estimate by introducing one additional communication round trip, which can subsequently reduce the data transferred in query processing. Experimental results show that both BulkDBPA and 4RUT require much less data transferred and response time than the competitors including Simple Algorithm and TPUT and each has its own suitable application environments respectively.**

***Index Terms*—top-*k* query, highly distributed, communication cost, BulkDBPA, 4RUT**

I. Introduction

Top-*k* queries have attracted much interest in many research areas such as network and system monitoring [1], data stream systems [2], information retrieval [3], sensor networks [4], peer-to-peer (P2P) systems [5][6], big data processing [7], etc., for they can avoid overwhelming the user with huge numbers of low quality results which are resource-consuming. Top-*k* queries retrieve the best matched *k* objects for users. The problem can be modeled as follows [8]: given a set O of n data objects O_i ($1 \le i \le n$), each object O_i is described by m attributes and has one score $s_j(O_i)$ on each attribute a_j ($1 \le j \le m$), and further define a scoring function $f(O_i) = f(s_1(O_i), s_2(O_i),\ldots, s_m(O_i))$ to compute the overall score of each object, and a top-*k* query requests to retrieve the k data objects whose overall scores are the highest among the set O.

Although much work has been done to solve the problem of top-*k* query processing in centralized database systems [9], the problem in distributed environments has not been well studied. Data generation, storage and application are becoming increasingly distributed. This paper focuses on top-*k* query processing in highly distributed environments such as P2P systems and sensor networks. The simplest solution for this problem is the Simple Algorithm (SA) [8], which collects the data of all nodes to the query initiator and processes the top-*k* query on the initiator in a centralized way. Since its high communication cost, SA is clearly inefficient. In highly distributed environments, the bandwidth among nodes is usually limited. Therefore, the key for a top-*k* query algorithm to be efficient is to have low communication cost. For certain distributed systems, the communication cost of an algorithm is mainly determined by two factors: data transferred and communication round trips. Data transferred represents the amount of data transferred among nodes and communication round trips denote the times of communication among nodes during the algorithm execution. In this paper, we try to propose efficient top-*k* query processing algorithms by limiting these two factors.

Starting from one of the key factors, data transferred, we have presented a novel algorithm called BulkDBPA (Bulk Distributed Best Position Algorithm) in our early publication [8]. BulkDBPA is derived from the centralized algorithm BPA2 [10] which is, to our best knowledge, the centralized top-*k* algorithm with lowest data access, benefit from the idea of best position and direct access mode. But a naive direct extension of BPA2 into highly distributed environments is infeasible. Although it will have the advantage of low data transferred, the communication round trips will be too huge, leading to worse query performance [8]. To solve this problem, BulkDBPA utilizes bulk read and bulk transfer mechanism for data access and transfer in order to reduce the communication round trips.

Considering the other key factor, communication round trips, we propose another novel algorithm called

Manuscript received January 29, 2013; revised April 30, 2013; accepted June 19, 2013.
This work is supported by National High-Tech R&D (863) Program of China (2010AA012400), National Natural Science Foundation of China (61272539) and the Foundation of Zhejiang Provincial Key Innovation Team on Sensor Networks of China (2009R50046).
Corresponding Author: Qiming Fang; fangqiming@gmail.com.

doi:10.4304/jcp.9.9.2000-2006

4RUT (4-Round-trip Uniform-Threshold) which is inspired by the distributed top-*k* algorithm TPUT (Three-Phase Uniform-Threshold) [11]. TPUT is a threshold-based algorithm and only takes three communication round trips to get the exact top-*k* results. TPUT consists of three steps: lower bound estimation, pruning and final lookup. Our proposed 4RUT tries to improve the lower bound estimate by introducing one additional communication round trip in order to reduce its data transferred in the later steps of pruning and final lookup so as to reduce its overall communication cost.

The new contributions of this paper are: firstly, a new efficient top-*k* query algorithm called 4RUT in highly distributed environments is proposed, which only takes four communication round trips to retrieve top-*k* results. Secondly, experiments are conducted to evaluate the performance of 4RUT as well as the early presented BulkDBPA. Experimental results show that both BulkDBPA and 4RUT require much less data transferred and response time than the competitors including SA and TPUT. Thirdly, this paper compares BulkDBPA and 4RUT on their performance and applicability for different environments. The two algorithms are both efficient and each has its own suitable application environments respectively.

Overall, we propose two efficient top-*k* query algorithms, BulkDBPA and 4RUT, by limiting the two major factors of communication cost in highly distributed environments, data transferred and communication round trips, respectively. So it can form an organic whole solution for the problem of top-*k* query processing in highly distributed environments over vertically partitioned data. For convenience of analysis and comparison as well as deeper understanding of the two novel algorithms, from their working mechanisms, features to effects, we will give a brief description of the early presented BulkDBPA algorithm and quote some representative experimental results in this paper.

The rest of this paper is organized as follows. Section II reviews the related work, and the BulkDBPA and 4RUT algorithms are presented in Section III and IV respectively. Section V gives the performance evaluation of the algorithms and finally we conclude in Section VI.

II. Related Work

In centralized systems, the most important top-*k* query algorithms include Fagin's Algorithm (FA) [12], Threshold Algorithm (TA) [13] and Best Position Algorithms (BPA and BPA2) [10]. TA, BPA and BPA2 are all threshold-based algorithms, and BPA2 has the lowest data access and the least query response time due to its best position and direct access mechanisms.

In distributed environments, the research on top-*k* query processing can be divided into two categories according to the data partitioning over nodes. One is over horizontally partitioned data in which each node stores a part of the data objects and their scores on all attributes. The other is over vertically partitioned data in which each node is responsible for one attribute, i.e. scores of all data objects on each attribute are stored in one node.

For horizontally partitioned data, the top-*k* query processing should involve all nodes in general and the proposed methods try to avoid this with the help of index [14], histogram-based routing filters [15], statistics-based heuristic [16], branch caching [17], skyline operation [18], etc., some of which only retrieve approximate results.

For vertically partitioned data, only the nodes with queried attributes will be involved in query processing and the research objective is to reduce the data access on each node and data transfer among nodes. Zhang and Suel [19] proposed DTA, a distributed extension of TA, and used Bloom Filter to improve its performance. Cao and Wang [11] presented TPUT algorithm which only requires three round trips to get the top-*k* results. Yu et al. [20] proposed algorithms similar to TPUT while using non-uniform threshold for each data list. TJA [21] is also a three-phase algorithm based on non-uniform threshold, and performs the score calculation in the network rather than in a centralized way. Akbarinia et al. [22] presented top-*k* query algorithms in Distributed Hash Tables (DHTs) based on their specific data storage mechanism. Besides, some algorithms retrieve only approximate top-*k* results, e.g. KLEE [23].

This paper focuses on the problem of exact top-*k* query processing over vertically partitioned data in highly distributed environments and tries to propose more efficient algorithms with lower communication cost.

III. BulkDBPA Algorithm

For convenience of problem description, we have the same assumption with [8] that in a distributed system of N nodes, each node V_i stores a data list L_i consisting of the n data objects and their scores on attribute a_i. Each data item in list L_i is a pair of $<O_j, s_i(O_j)>$ which represents the data object O_j has a local score of $s_i(O_j)$ in list L_i. Each list is sorted in descending order of the local scores and the scoring function f is a monotone increasing function.

BPA2 is a centralized algorithm taking very low data access due to its best position and direct access mechanisms. The best position of a list is the greatest seen position in the list such that any lower position is also seen in data access. Direct access is a mode of access that reads the data object at a given position in a list.

We plan to extend BPA2 into distributed environments, expecting to develop an algorithm with very low data transferred among nodes. But naive distributed extension will lead to a poor algorithm which needs a large number of communication round trips, shown by DBPA [8]. So a novel algorithm called BulkDBPA has been proposed [8], which employs bulk read and bulk transfer mechanism to reduce the communication round trips. It reads a bulk of unseen data items closely after the best position of each queried list at each direct access and sends the bulk of data items to the query initiator at one time.

The BulkDBPA algorithm works as follows [8]:

1. The query initiator R finds out the set of nodes V which store the lists involved in the query. For each list L_i on node V_i, set its best position $bp_i = 0$.
2. In parallel, do direct access to each list L_i, read B

unseen data items closely after position bp_i and send them to R. All the returned data objects form a set D.
3. In parallel, do random access to each list L_i to find the local score of each data object in D and send it to R.
4. If a direct access or random access to each list L_i changes its best position, then update bp_i and send $s_i(bp_i)$ to R along with the local score sent in step 3.
5. On node R, compute the overall score of each data object in set D and maintain in a set Y the k seen data objects whose overall scores are the highest among all the data objects seen so far.
6. On node R, compute the threshold $T = f(s_1(bp_1), s_2(bp_2), \ldots, s_m(bp_m))$. If Y has involved k data objects whose overall scores are not lower than T, then stop doing direct access. Otherwise, go to step 2.
7. Return Y. BulkDBPA finished.

Different settings of bulk size B will affect the query performance. The bulk size B can be predefined in the query system or determined by the query initiator, and can also be set for each query separately. In general, BulkDBPA can employ two kinds of bulk size strategies: fixed size and variable size. The former uses a fixed bulk size for all different queries and the latter uses different bulk sizes for different queries [8].

IV. 4RUT Algorithm

TPUT is a most concerned distributed top-k query algorithm. It is a threshold-based algorithm which reduces communication cost by pruning away ineligible data objects and terminates in three communication round trips between the query initiator and the queried nodes to get the exact top-k results regardless of datasets and queries. This characteristic avoids TPUT to suffer from the problem of too many or uncertain number of communication round trips which will lead to too much communication cost or introduce uncertainty to the algorithm efficiency. TPUT has good performance in reducing communication cost, but we believe it is not perfect yet. We found out through theoretical and experimental analysis that the communication cost of TPUT can be further reduced and finally we propose a novel and more efficient algorithm called 4RUT.

TPUT executes in three steps: lower bound estimation, pruning and final lookup. Firstly, it determines a lower bound estimate for the k'th value; then it uses the estimate to prune away ineligible data objects as much as possible; finally, it looks up the resulting set of objects in all queried nodes to identify the top-k results. In the first step, each queried node sends the top k items in its list to the query initiator, then the query initiator calculates the partial scores of the collected objects and takes the k'th highest partial score as the top-k lower bound estimate, which is used to guide the pruning step. We think if we can improve the lower bound estimate, we can certainly reduce the communication cost in the later steps, because better lower bound estimate can help to prune away more ineligible data objects and thus less data objects are needed to be sent to the query initiator. Based on this idea, we propose 4RUT, introducing one additional communication round trip to obtain better lower bound estimate. 4RUT executes in four steps:

1. The query initiator R finds out the set of nodes V which store the lists involved in the query. Each node V_i sends the top k items in lists L_i to R, and all the returned data objects form a set D.
2. R collects all unread local scores for each object in D from each list L_i, calculates the overall score of each object in D, and takes the k'th highest score as the top-k lower bound estimate.

3 and 4. The same as the step 2 and 3 of TPUT.

We can see that in step 1 and 2 of 4RUT, we use two communication round trips to get the exact overall score of each object in D rather than partial score in TPUT. The exact overall score is certainly not worse than (and usually is better than) its partial score for each data object. Therefore, this method can be expected to obtain better lower bound estimate which will lead to less communication cost in later steps of the algorithm. We will compare the performance of 4RUT with TPUT in next section.

V. Experimental Evaluation

A. Experimental Settings

We use the same experimental settings with [8] shown in Table I. The hardware environments are a LAN with 100Mbps bandwidth and PCs with Intel Xeon 2.66GHz CPU and 1GB memory. Five top-k query algorithms SA+BPA2, DBPA, BulkDBPA, TPUT and 4RUT are implemented using Java programming language in FreePastry system, an open source implementation of the famous P2P protocol Pastry [24] which can provide $O(\log N)$-hops routing in an N-nodes system. SA+BPA2 represents the SA algorithm based on BPA2, i.e. the query initiator collects all the data from involved nodes and then solves the top-k query using BPA2. DBPA [8] represents the naive distributed extension of BPA2. The same three metrics with [8] are used to evaluate the performance of the algorithms: query response time, data transferred and communication round trips.

TABLE I
Experimental Parameter Settings [8]

Parameter	Value Range	Default Value
Number of nodes, N	16	16
Number of data objects, n	1000-1000000	100000
Number of queried lists, m	2-10	4
Dataset	Uniform, Gaussian	Uniform
k	1-100	10

B. Experimental Results

The experimental results shown in this subsection are the average results of 10 runs.

1) Comparison of different algorithms

Figure 1 shows the performance comparison of the five algorithms over Uniform and Gaussian datasets, where BulkDBPA employs the uniform bulk size strategy setting B=k for each query. We can see that DBPA and

BulkDBPA both have less data transferred than SA+BPA2 and TPUT over the two datasets. This is because they both absorb the advantage of low data access of the centralized algorithm BPA2. But due to its much more communication round trips, the response time of DBPA is not acceptable at all. BulkDBPA improves its response time by utilizing bulk read and bulk transfer mechanism which significantly reduces its communication round trips. For 4RUT, although it takes one more round trip than TPUT (4 vs. 3), it outperforms TPUT on data transferred as well as response time, especially on the Gaussian dataset. Furthermore, we can see that the performance of 4RUT is comparable with BulkDBPA.

2) Effect of n and m

Figure 2 shows the performance of BulkDBPA, TPUT and 4RUT while varying the parameters *n* and *m*. It can be seen that the data transferred and response time increase with *n* and *m* for each of the algorithms. This is a natural result since the workload of each algorithm will increase with *n* and *m*. Another observation is that the performance of BulkDBPA and 4RUT are both better than TPUT on any parameter setting, proving the efficiency of the new proposed algorithms.

3) Bulk size strategy of BulkDBPA

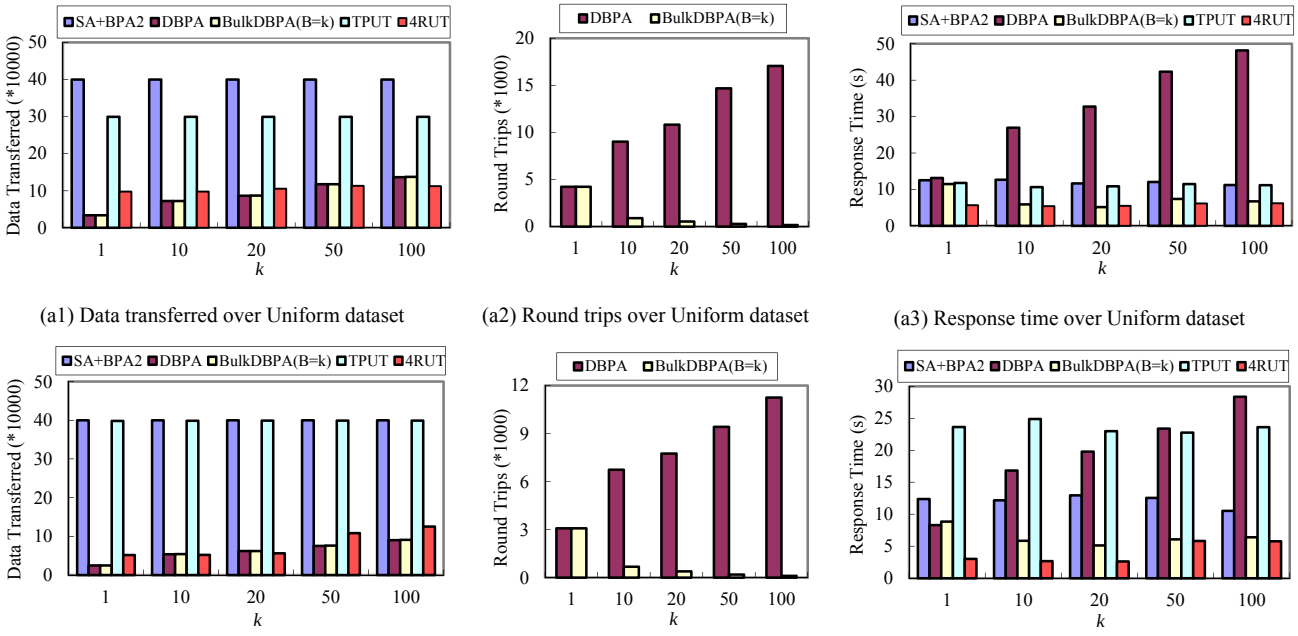
The effect of different bulk size strategies for BulkDBPA is investigated and the results are shown in Figure 3. The strategies used in experiments are listed in Table II [8] where *step* denotes the step of direct access in BulkDBPA. We can conclude that the simple fixed strategy can achieve reasonable performance if we can determine an appropriate bulk size and among the variable strategies, the linear strategy can be considered as the best.

TABLE II.
THE BULK SIZE STRATEGIES USED IN EXPERIMENTS[8]

Strategy		Expression
Fixed		$B = B_0$
Variable	Uniform	$B = k$
	Linear	$B = k \times step$
	Pow	$B = k \times step^2$
	Log	$B = k \times \log_2(step+1)$
	Sqrt	$B = k \times \sqrt{step}$
	Exp	$B = k \times 2^{step}$

4) Comparison of BulkDBPA and 4RUT

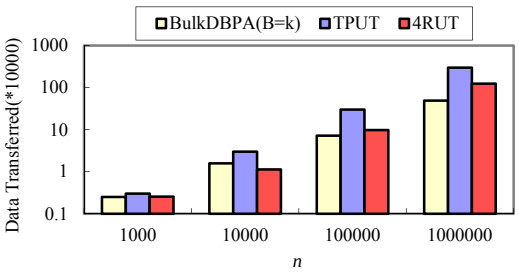
From the conclusion of last subsection, the linear strategy can be considered as the best bulk size strategy for BulkDBPA among the candidates. So we are interested in comparing the performance of BulkDBPA employing linear strategy with 4RUT. Figure 4 shows the results of response time on different settings. From these results, we cannot determine the exact winner from BulkDBPA and 4RUT, because each competitor has its own advantages for different settings. For example, BulkDBPA requires less response time than 4RUT when $k \leq 50$ on the setting of n=100000 and m=4 over the Uniform dataset, but 4RUT becomes the winner when k=100. But we can say in general that BulkDBPA is more suitable for lower m and k which requires relatively less query workload, while 4RUT is more efficient when m or k is higher.



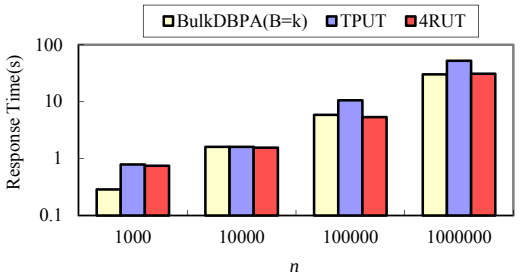
(a1) Data transferred over Uniform dataset (a2) Round trips over Uniform dataset (a3) Response time over Uniform dataset

(b1) Data transferred over Gaussian dataset (b2)Round trips over Gaussian dataset (b3) Response time over Gaussian dataset

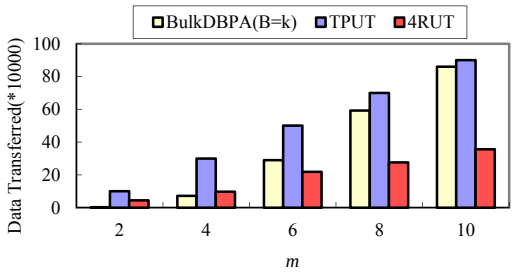
Figure 1. Performance comparison of different algorithms (m=4, n=100000).



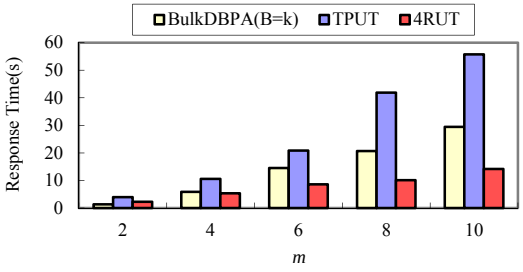
(a1) Data transferred over Uniform dataset (k=10, m=4)



(a2) Response time over Uniform dataset (k=10, m=4)

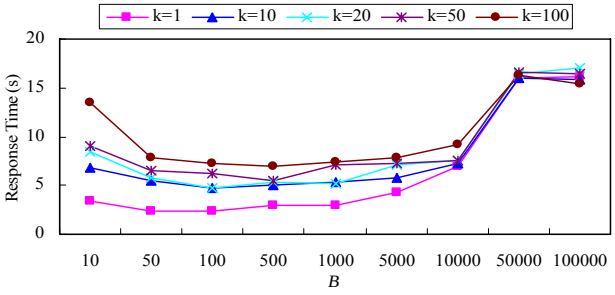


(b1) Data transferred over Gaussian dataset (k=10, n=100000)

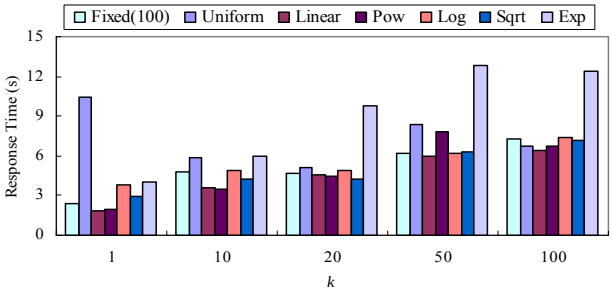


(b2) Response time over Gaussian dataset (k=10, n=100000)

Figure 2. Performance of BulkDBPA and 4RUT while varying n and m

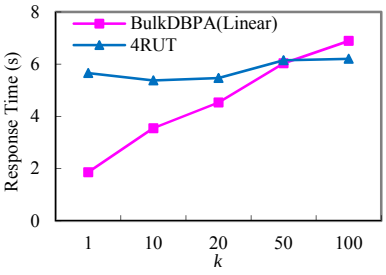


(a) Fixed bulk size

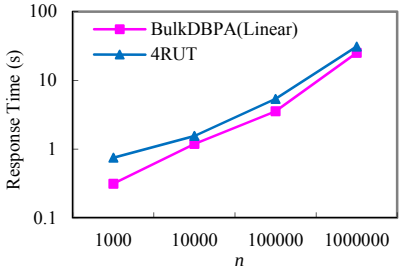


(b) Different bulk size strategies

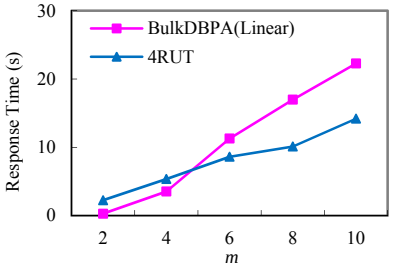
Figure 3. Response time of BulkDBPA with different bulk size strategies over Uniform dataset (m=4, n=100000) [8] .



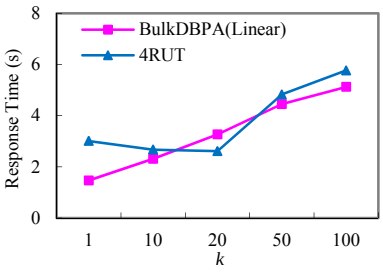
(a1) Varying k over Uniform dataset (m=4, n=100000)



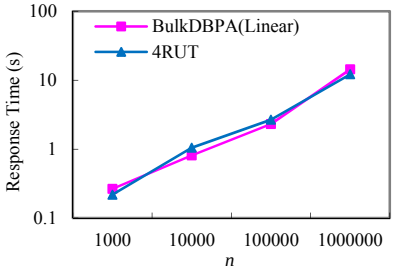
(a2) Varying n over Uniform dataset (k=10, m=4)



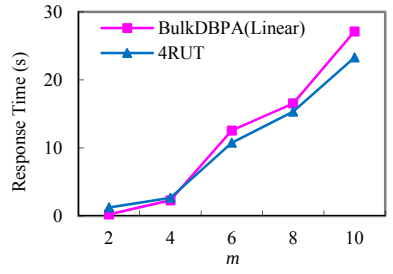
(a3) Varying m over Uniform dataset (k=10, n=100000)



(b1) Varying k over Gaussian dataset (m=4, n=100000)



(b2) Varying n over Gaussian dataset (k=10, m=4)



(b3) Varying m over Gaussian dataset (k=10, n=100000)

Figure 4. Comparison of BulkDBPA and 4RUT on response time over Uniform and Gaussian datasets.

VI. Conclusion

This paper proposes two algorithms, BulkDBPA and 4RUT, for top-*k* query processing in highly distributed environments over vertically partitioned data. Communication cost is the key factor to affect the query performance in these environments. Our effort is put on limiting the data transferred and communication round trips to reduce the communication cost. Absorbing the idea of best position from one of the best centralized algorithms BPA2, BulkDBPA gets the advantage of low data transferred. It further employs bulk read and bulk transfer mechanism to reduce the communication round trips. 4RUT is a 4-round-trip threshold-based algorithm. It improves the 3-round-trip algorithm TPUT by introducing one additional communication round trip to obtain a better top-*k* lower bound estimate, which subsequently reduces the communication cost of the algorithm. Experimental results show that both BulkDBPA and 4RUT require much less data transferred and response time than other competitors and each has its own suitable application environments respectively. Future work includes detailed investigation of more different bulk size strategies for BulkDBPA and further optimizations such as using Bloom Filter to reduce the data transferred.

References

[1] B. Babcock and C. Olston, "Distributed Top-k Monitoring," *Proc. SIGMOD Conf.*, ACM Press, Jun. 2003, pp. 28-39.

[2] A. Metwally, D. Agrawal, and A. E. Abbadi, "An Integrated Efficient Solution for Computing Frequent and Top-k Elements in Data Streams," *J. ACM Trans. Database Systems (TODS)*, vol. 31, no. 3, 2006, pp. 1095-1133.

[3] T. Tran, H. Wang, S. Rudolph, and P. Cimiano, "Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data," *Proc. IEEE Int. Conf. Data Engineering (ICDE)*, IEEE Press, 2009, pp. 405-416.

[4] P. Andreou, D. Zeinalipour-Yazti, M. Vassiliadou, P. K. Chrysanthis, and G. Samaras, "KSpot: Effectively Monitoring the K Most Important Events in a Wireless Sensor Network," *Proc. IEEE Int. Conf. Data Engineering (ICDE),* IEEE Press, 2009, pp. 1503-1506.

[5] G. Weikum, "Peer-to-peer web search: euphoria, achievements, disillusionment, and future opportunities," *Proc. From Active Data Management to Event-Based Systems and More*, LNCS 6462, 2010, pp. 195-208.

[6] I. Chrysakis, C. Chalkidis, D. Plexousakis, "Evaluation of top-k queries in peer-to-peer networks using threshold algorithms," *Proc. 19th ACM Int. Conf. Information and Knowledge Management (CIKM)*, ACM New York, 2010, pp.1305-1308.

[7] Y.J. Sun, Y. Yuan, G.R. Wang, "Top-k query processing over uncertain data in distributed environments," *J. World Wide Web,* vol.15, no.4, 2012, pp.429-446.

[8] Q.M. Fang, Y. Zhao, G.W. Yang, B. Wang, and W.M. Zheng, "Best Position Algorithms for Top-k Query Processing in Highly Distributed Environments," Proc. 1st Int. Conf. Networking and Distributed Computing (ICNDC), IEEE, 2010, pp.397-401.

[9] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A Survey of Top-k Query Processing Techniques in Relational Database Systems," *ACM Comput. Surv.*, vol. 40, no. 4, Oct. 2008, pp. 1-58.

[10] R. Akbarinia, E. Pacitti, and P. Valduriez, "Best Position Algorithms for Top-k Queries," *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2007, pp. 495-506.

[11] P. Cao and Z. Wang, "Efficient Top-k Query Calculation in Distributed Networks," *Proc. Annual ACM Symp. Principles of Distributed Computing (PODC)*, 2004, pp. 206-215.

[12] R. Fagin, "Combining Fuzzy Information From Multiple Systems," *J. Comput. System Sci.*, vol. 58, no. 1, 1999, pp. 83-99.

[13] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," *Proc. Symp. Principles of Database Systems (PODS)*, 2001, pp. 102-113.

[14] W. T. Balke, W. Nejdl, W. Siberski, and U. Thaden, "Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks," *Proc. IEEE Int. Conf. Data Engineering (ICDE)*, IEEE Press, 2005, pp. 174–185.

[15] K. Hose, M. Karnstedt, K. U. Sattler, and D. Zinn, "Processing Top-N Queries in P2P-based Web Integration Systems with Probabilistic Guarantees," *Proc. Int. Workshop Web and Databases (WebDB)*, 2005, pp. 109-114.

[16] R. Akbarinia, E. Pacitti, and P. Valduriez, "Reducing Network Traffic in Unstructured P2P Systems Using Top-k Queries," *Distributed and Parallel Databases*, vol. 19, no. 2-3, May. 2006, pp. 67-86.

[17] K. Zhao, Y. Tao, and S. Zhou, "Efficient Top-k Processing in Large-Scaled Distributed Environments," *Data and Knowledge Engineering (DKE)*, vol. 63, no. 2, 2007, pp. 315-335.

[18] A. Vlachou, C. Doulkeridis, K. Norvag, and M. Vazirgiannis, "On Efficient Top-k Query Processing in Highly Distributed Environments," *Proc. SIGMOD Conf.*, ACM, 2008, pp. 753-764.

[19] J. Zhang and T. Suel, "Efficient Query Evaluation on Large Textual Collections in a Peer-to-Peer Environment," *Proc. Peer-to-Peer Computing (P2P)*, 2005, pp. 225-233.

[20] H. Yu, H. G. Li, P. Wu, D. Agrawal, and A. E. Abbadi, "Efficient Processing of Distributed Top-k Queries," *Proc. Database and Expert Systems Applications. (DEXA)*, 2005, pp. 65-74.

[21] D. Zeinalipour-Yazti, Z. Vagena, V. Kalogeraki, D. Gunopulos, V. J. Tsotras, M. Vlachos, N. Koudas, and D. Srivastava, "Finding the K Highest-Ranked Answers in a Distributed Network," *Comput. Networks*, vol. 53, 2009, pp. 1431-1449.

[22] R. Akbarinia, E. Pacitti, and P. Valduriez, "Processing Top-k Queries in Distributed Hash Tables," *Proc. Euro-Par Conf.*, Springer-Verlag, 2007, pp. 489-502.

[23] S. Michel, P. Triantafillou, and G. Weikum, "KLEE: a Framework for Distributed Top-k Query Algorithms," *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2005, pp. 637-648.

[24] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. IFIP/ACM Int. Conf. Distributed Systems Platforms (Middleware)*, Springer-Verlag, 2001, pp. 329-350.

Qiming Fang was born in Zhejiang Province, China in 1982. He received his Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China in 2010. He is now working as a lecturer in Hangzhou Dianzi University, Hangzhou, China. His research is focused on distributed systems, information retrieval, big data processing, etc. He is a member of China Computer Federation, IEEE Computer Society and ACM.

Guangwen Yang was born in Shanxi Province, China in 1964. He received his B.S. degree in applied mathematics from Taiyuan University of Technology, Taiyuan, China in 1984, M.S. degree in applied mathematics and Ph.D. degree in computer architecture from Harbin Institute of Technology, Harbin, China in 1987 and 1996 respectively. He is now a professor in Department of Computer Science and Technology as well as Center for Earth System Science, Tsinghua University, Beijing, China. His research interests include high performance computing, algorithm design and analysis, grid and cloud computing, earth system model, etc. He is a member of China Computer Federation, IEEE and ACM. He has taken charge of many research projects, finished over 60 publications and won several research awards including the second prize of National Scientific and Technological Progress Award of China.