Data Anonymization Using Map Reduce on Cloud based A Scalable Two-Phase Top-Down Specialization

Zorige Priyanka
M.Tech (CSE) Student
GietEnggcollege
Rajahmundry,
Andhra Pradesh, India
Vamsi.priyanka5@gmail.com

K Nagaraju
Associate professor
Department of Cse
GietEnggcollegeRajahmundry
Andhra Pradesh, India
nagarajuknr.k@gmail.com

Dr. Y Venkateswarlu
Professor & Head
Department of CSE
GIET Engg College,
Rajahmundry,
Andhra Pradesh, India
yalla_venkat@yahoo.com

Abstract—A large number of cloud services require users to impart` private data like electronic health records for data analysis or Mining, bringing privacy concerns. Anonymizing information sets through generalization to fulfill certain security prerequisites, for example, k-anonymity is a broadly utilized classification of protection safeguarding procedures At present, the scale of information in numerous cloud applications increments immensely as per the Big Data pattern, in this manner making it a test for normally utilized programming instruments to catch, oversee, and process such substantial scale information inside a bearable slipped by time. As an issue, it is a test for existing anonymization methodologies to accomplish security protection on security touchy extensive scale information sets because of their inadequacy of adaptability. In this paper, we propose a versatile two-stage top-down specialization (TDS) methodology to anonymize huge scale information sets utilizing the Map reduce schema on cloud. Experimental evaluation results demonstrate that with our approach, the scalability and efficiency of TDS can be significantly improved over existing approaches.

Index Terms: *Data anonymization, top-down specialization, MapReduce, cloud, privacy preservation*

*****

I. INTRODUCTION:

At present Cloud computing, a disruptive trend, poses a Significantbrunt on current IT industry and investigate communities [1], [2], [3]. Cloud computing provides enormous computation power and storage capacity via utilizing a large number of commodity computers together, enabling users to deploy applications cost-effectively without heavy infrastructure investment. Cloud users can reduce huge upfront investment of IT infrastructure, and concentrate on their own core business.

Data anonymization has been extensively studied and widely adopted for data privacy preservation in non interactive data publishing and sharing scenarios [4]. Data anonymization refers to hiding identity and/or sensitive data for owners of data records. Then, the privacy of an individual can be effectively preserved while certain aggregate information is exposed to data users for diverse analysis and mining. A variety of anonymization algorithms with different anonymization operations have been proposed [5], [6], [7], [8]. However, the scale of data sets that need anonymizing in some cloud applications increases extremely in accordance with the cloud computing and Big Data trends [1], [9]. Data sets have become so large that anonymizing such data sets is becoming a considerable challenge for traditional anonymization algorithms. The researchers have begun to investigate the scalability problem of large-scale data anonymization [10], [11].

Large-scale data processing frameworks like MapReduce [12] have been integrated with cloud to provide powerful computation capability for applications. So, it is promising to adopt such frameworks to address the scalability problem of anonymizing large-scale data for privacy preservation. In our research, we leverage MapReduce, a widely adopted parallel data processing framework, to address the scalability problem of the top-down specialization (TDS) approach [12] for large-scale data anonymization. The TDS approach, offering a good tradeoff between data utility and data consistency, is widely applied for data anonymization [5], [13], [14], [15]. Most TDS algorithms are centralized, resulting in their inadequacy in handling largescale data sets. Although some distributed algorithms have been proposed [13], [15], they mainly focus on secure anonymization of data sets from multiple parties, rather than the scalability aspect. As the MapReduce computation paradigm is relatively simple, it is still a challenge to design proper MapReduce jobs for TDS

In this paper, we propose a highly scalable two-phase TDS approach for data anonymization based on MapReduce on cloud. To make full use of the parallel capability of MapReduce on cloud, specializations required in an anonymization process are split into two phases. In the first one, original data sets are partitioned into a group of smaller data sets, and these data sets are anonymized in parallel, producing intermediate results. In the second one, the intermediate results are integrated into one, and further anonymized to achieve consistent k-anonymous [23] data sets. We evaluate our approach by conducting experiments on real-world data sets. Experimental results demonstrate that with our approach, the scalability and efficiency of TDS can be improved significantly over existing approaches.

This paper is organized as follows: In Section 2, formulates the two-phase TDS approach, and Section We empirically evaluate our approach in Section 3. Finally, we conclude this paper and discuss future work in Section 4.

II. TWO-PHASE TOP-DOWN SPECIALIZATION (TPTDS)

The sketch of the TPTDS approach explained in section 2.1. the TPTDS has three components namely, data partition, anonymization level merging, and data specialization are detailed in Sections 2.2, 2.3, and 2.4, respectively.

2.1 Sketch of Two-Phase Top-Down Specialization

We propose a TPTDS approach to conduct the computation required in TDS in a highly scalable and efficient fashion. The two phases of our approach are based on the two levels of parallelization provisioned by MapReduce on cloud. Basically, MapReduce on cloud has two levels of parallelization, i.e., job level and task level. Job level parallelization means that multiple MapReduce jobs can be executed simultaneously to make full use of cloud infrastructure resources. Combined with cloud, MapReduce becomes more powerful and elastic as cloud can offer infrastructure resources on demand, for example, Amazon Elastic MapReduce service [16]. Task level parallelization refers to that multiple mapper/reducer tasks in a MapReduce job are executed simultaneously over data splits. To achieve high scalability, we parallelizing multiple jobs on data partitions in the first phase, but the resultant anonymization levels are not identical. To obtain finally consistent anonymous data sets, the second phase is necessary to integrate the intermediate results and further anonymized entire data sets. Details are formulated as follows.

In the first phase, an original data set D is partitioned into smaller ones. Let Di, 1 ≤ i≤ p, denote the data sets partitioned from D the, where p is the number of partitions.

$$\sum_{i=1}^{p} D_i , D_i \cap D_j , D_j \ = \ \emptyset, 1 \ \leq i \ \leq \mathrm{p}$$

The details of how to partition D will be discussed in Section 2.2. Then, we run a subroutine over each of the partitioned data sets in parallel to make full use of the job level parallelization of MapReduce. The subroutine is a MapReduce version of centralized TDS (MRTDS) which concretely conducts the computation required in TPTDS. Algorithm 1 depicts the sketch of the two-phase TDS approach.

Algorithm 1. Sketch Of Two-Phase TDS (TPTDS).

Input: Data set D, anonymity parameters k, kI and the number of partitions p.

Output: Anonymous data set D*

1. Partition D into Di,1 _ i _ p.
2. Execute MRTDS(Di, kl, ALo) $\rightarrow$AL1i, 1 ≤ i ≤ p in parallel as multiple MapReduce jobs.
3. Merge all intermediate anonymization levels into one merge(ALi1, ALi2,……. ALip) $\rightarrow$ AL1
4. Execute MRTDS(D, k, ALI) $\rightarrow$ AL* to achievek-anonymity.
5. Specialize D according to AL* Output D*

In the Partition Step, a data record here can be treated as a point in an m-dimension space, where m is the number of attributes. Thus, the intermediate anonymization levels derived fromDi, 1 ≤i ≤ p, can be more similar so that we can get a better merged anonymization level. Random sampling technique is adopted to partition D, which can satisfy the above requirement. Specifically, a random number rand, 1 ≤ rand≤p, is generated for each data record. A record is assigned to the partition Drand. Algorithm 2 shows the MapReduce program of data partition. Note that the number of Reducers should be equal to p, so that each Reducer handles one value of rand, exactly producing p resultant files. Each file contains a randomsample of D. the data partition algorithm is shown below

Algorithm 2. Data Partition Map & Reduce.

Input: Data record (IDr, r), r € D, partition parameter p.

Output: Di, 1 ≤i≤ p.

Map: Generate a random number rand,where 1 ≤ rand ≤ p; emit (rand, r).

Reduce: For each rand, emit (null, list(r)).

Once partitioned data sets Di, 1 ≤ i ≤ p. are obtained, werun MRTDS(D, k, ALo) on these data sets in parallel toderive intermediate anonymization levels AL*I, 1 ≤ i ≤ p

Then, we run a subroutine over each of the partitioned data sets in parallel to make full use of the job level parallelization of MapReduce In the second step, all intermediate anonymization levels are merged into one. The merged anonymization level is denoted as ALI . The merging process is formally represented as function merge(AL’1, AL’2, ………., AL’p} -> AL’ where AL’ denotes the final anonymization level

Data Specialization

An original data set D is concretely specialized for anonymization in a one-pass MapReduce job. After obtaining the merged intermediate anonymization level ALI, we run MRTDS(D, k,AL’) on the entire data set D, and get the final anonymization level AL*. Then, the data set D, is anonymized by replacing original attribute values in D with the responding domain values in AL*. Details of Map and Reduce functions of the data specialization MapReduce job are described in Algorithm 3. The Map function emits anonymous records and its count. The Reduce function simply aggregates these anonymous records and counts their number. An anonymous record and its count represent a QI-group. The QI-groups constitute the final anonymous data sets

Algorithm 3. Data Specialization Map & Reduce.

Input: Data record (IDr, r), r € D.; Anonymization level AL*.

Output: Anonymous record (r*, count).

Map: Construct anonymous record r* = p1 (p2, p3 . . . pm,sv), pi, 1 ≤ i≤ m, is the parent of a specialization in currentAL and is also an ancestor of vi in r1 emit (r*, count).

Reduce: For each r*, sum = ∑ count; emit (r*, sum).

III. RESULTS AND DISCUSSIONS

To evaluate the effectiveness and efficiency of our two phase approach, we compare it with the centralized TDS approach proposed in [5], denoted as CentTDS. CentTDS is the state-of-the-art approach for TDS anonymization. Scalability and data utility are considered for the effectiveness. For scalability, we check whether both approaches can still work and scale over large-scale data sets. Data utility is measured by the metric ILoss, a general purpose data metric proposed in [17]. Literally, ILoss means information loss caused by data anonymization. Basically, higher ILoss indicates less data utility. The execution time of CentTDS and TPTDS are denoted as TCent and TTP, respectively.

The overheads of our approach are mainly introduced by the MapReduce built-in operations and the parallelization in the first phase of TPTDS. Built-in MapReduce operations like data splitting and key-value pair sorting and transmission will cause overheads. The overheads are hard to quantitatively measure as they are implementation-, configuration-, and algorithm-specific. The extra specializations in the first phase incur overheads affecting the efficiency of TPTDS heavily.

Experiment Evaluation: Experiment Settings

Our experiments are conducted in a cloud environment named U-Cloud. U-Cloud is a cloud computing environment. The system overview of U-Cloud has been depicted in Fig. 2. The computing facilities of this system are located among several labs at UTS. On top of hardware and Linux operating system (Ubuntu), we install KVM virtualization software [18] that virtualizes the infrastructure and provides unified computing and storage resources. To create virtualized data centers, we install OpenStack open source cloud environment [19] for global management, resource scheduling and interaction with users. Further, Hadoop [20] clusters are built based on the OpenStack cloud platform to facilitate large-scale data processing.

We use Adult data set [21], a public data set commonly used as a de facto benchmark for testing anonymization algorithms [6], [13]. We generate data sets by enlarging the Adult data set according to the approach in [13].Both TPTDS and CentTDS are implemented in Java. Further, TPTDS is implemented with standard Hadoop MapReduce API and executed on a Hadoop cluster built on OpenStack. The k-anonymity parameter is set as 50 throughout all experiments. Each round of experiment is repeated 20 times. The mean of the measured results is regarded as the representative.

We conduct experiments to evaluate the effectiveness and efficiency of our approach. In this we compare TPTDS with CentTDS from the perspectives of scalability and efficiency.

In this proposed method, we measure the change of execution time TCent and TTP with respect to S when p = 1. The size S varies from 50 MB to 2.5 GB. The 2.5 GB data set contains nearly 2:5 x 107 data records. The scale of data sets in our experiments is much greater than that in [5] and [13]. Thus, the data sets in our experiments are big enough to evaluate the effectiveness of our approach in terms of data volume or the number of data records. Note that ILCent= ILTP because TPTDS is equivalent to MRTDS when p = 1. So, we just demo the results of execution time. The results of proposed method are listed in Fig.2 and 3.

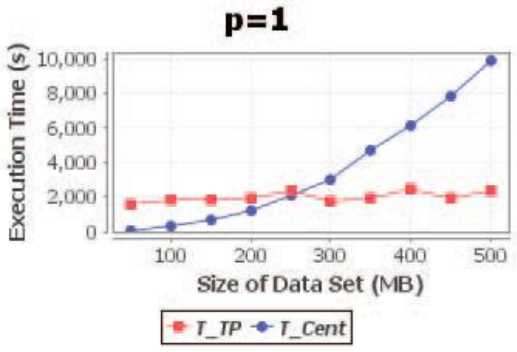

Figure2 : Change of execution time with respect to data size: TPTDS versus CentTDS.

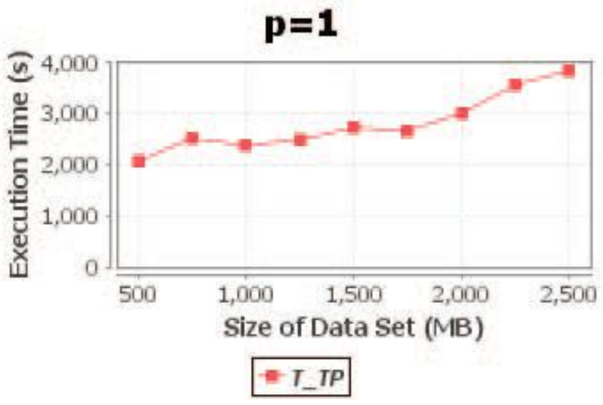

Figure 3: Change of execution time with respect to data size: TPTDS versus CentTDS.

Fig. 2 shows the change of TTP and TCent with respect to the data size ranging from 50 to 500 MB. From Fig. 2, we can see that both TTP and TCent go up when data size increases although some slight fluctuations exist. The fluctuations are mainly caused by the content of data sets. TCent surges from tens of seconds to nearly 10,000 seconds, while TTP increase slightly. The dramatic increase of TCent illustrates that the overheads incurred by maintaining linkage structure and updating statistic information rise considerably when data size increases. Before the point S= 250 MB, TTP is greater than TCent. But after the point, TTP is greater than TCent, and the difference between TCent and TTP becomes larger and larger with the size of data sets increasing. The trend of TTP and TCent indicates that TPTDS becomes more efficient compared with CentTDS for largescale data sets.

In our experiments, CentTDS fails due to insufficientmemory when the size of data set is greater than 500 MB.Hence, CentTDS suffers from scalability problem for largescaledata sets. To further evaluate the scalability andefficiency of TPTDS, we run TPTDS over data sets withlarger sizes. Fig 3 shows the change of TTP with respect tothe data size ranging from 500 MB to 2.5 GB. It can be seenfrom Fig. 3 that TTP grows linearly and stably with respectto the size of data sets. Based on the tendency of TTP, wemaintain that TPTDS is capable of scaling over large-scaledata sets efficiently.

The above experimental results demonstrate that ourapproach can significantly improve the scalability andefficiency compared with the state-of-the-art TDS approachwhen anonymizing large-scale data sets.

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we have investigated the scalability problem of large-scale data anonymization by TDS, and proposed a highly scalable two-phase TDS approach using MapReduce on cloud. Data sets are partitioned and anonymized in parallel in the first phase, producing intermediate results. Then, the intermediate results are merged and further anonymized to produce consistent k-anonymous data sets in the second phase. We have creatively applied MapReduce on cloud to data anonymization and deliberately designed a group of innovative MapReduce jobs to concretely accomplish the specialization computation in a highly scalable way. Experimental results on real-world data sets have demonstrated that with our approach, the scalability and efficiency of TDS are improved significantly over existing approaches. In cloud environment, the privacy preservation for data analysis, share and mining is a challenging research issue due to increasingly larger volumes of data sets, thereby requiring intensive investigation. We will investigate the adoption of our approach to the bottom-up generalization algorithms for data anonymization. Based on the contributions herein, we plan to further explore the next step on scalable privacy preservation aware analysis and scheduling on large-scale data sets. Optimized balanced scheduling strategies are expected to be developed towards overall scalable privacy preservation aware data set scheduling.

V. REFERENCES:

[1] S. Chaudhuri, “What Next?: A Half-Dozen Data Management Research Goals for Big Data and the Cloud,” Proceedings 31st Symposium on “Principles of Database Systems (PODS), pages:. 1-4, in the year 2012.

[2] M. Armbrust, A.D. Joseph, R. Katz A. Fox, R. Griffith, , “A View of Cloud Computing,” Comm. ACM, Volume: 53, Number: 4, Pages: 50-58, 2010.

[3] L. Wang, J. Zhan, Y. Liang and W. Shi, “In Cloud, Can Scientific Communities Benefit from the Economies of Scale?,” IEEE Transactions on Parallel and Distributed Systems, Volume: 23, Number: 2, Pages:296-303, Feb. 2012.

[4] B.C.M. Fung, R. Chen, K. Wang, and P.S. Yu, “Privacy-Preserving Data Publishing: A Survey of Recent Devel-opments,” ACM Computing Surveys, volume. 42, Number: 4, Pages: 1-53, 2010.

[5] B.C.M. Fung, K. Wang, and P.S. Yu, “Anonymizing Classification Data for Privacy Preservation,” IEEE Trans. Knowledge and Data Eng., Volume: 19, Number: 5, Pages: 711-725, May 2007.

[6] X. Xiao and Y. Tao, “Anatomy: Simple and Effective Privacy Preservation,” Proc. 32nd Int’l Conf. Very Large Data Bases (VLDB ’06), Pages: 139-150, 2006.

[7] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, “Incognito: Efficient Full-Domain K-Anonymity,” Proc. ACM SIGMOD Int’l Conf. Management of Data (SIGMOD ’05), Pages: 49-60, 2005.

[8] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, “Mondrian Multidimensional K-Anonymity,” Proc. 22nd Int’l Conf. Data Eng. (ICDE ’06), 2006.

[9] V. Borkar, M.J. Carey, and C. Li, “Inside ‘Big Data Management’: Ogres, Onions, or Parfaits?,” Proc. 15th Int’l Conf. Extending Database Technology (EDBT ’12), Pages: 3-14, 2012.

[10] LeFevre, DeWitt, and Ramakrishnan, “Workload-Aware Anonymization Techniques for Large-Scale Data Sets,” ACM Trans. Database Systems, Volume: 33, Number: 3, Pages: 1-47, 2008.

[11] Iwuchukwu and Naughton, “K-Anonymization as Spatial Indexing: Toward Scalable and Incremental Anonymization,” Proceedings of 33rd International Conference on Very Large Data Bases (VLDB ’07), Pages: 746-757, in the year 2007.

[12] Dean and Ghemawat, “Mapreduce: Simplified Data Processing on Large Clusters,” Communication in ACM, volume. 51, number . 1, Pages: 107-113, in the year 2008.

[13] Mohammed, Fung,. Hung P.C.K, and Lee CK, “Centralized and Distributed Anonymization for High-Dimensional Healthcare Data,” ACM Trans. Knowledge

Discovery from Data, Volume: 4, Number: 4, Article 18, 2010.

[14] Fung, Wang, Wang L, and. Hung P.C.K, “Privacy Preserving Data Publishing for Cluster Analysis,” Data and Knowledge Eng., Volume: 68, Number: 6, Pages: 552-575, in the year 2009.

[15] Mohammed, Fung, and Debbabi, “Anonymity Meets Game Theory: Secure Data Integration with Malicious Participants,” VLDB J., Volume: 20, Number: 4, Pages: 567-588, 2011.

[16] Amazon Web Services, “Amazon Elastic Mapreduce,” http:// aws.amazon.com/elasticmapreduce/, 2013.

[17] Xiao and Tao, “Personalized Privacy Preservation,” Proceedings of ACM SIGMOD Intternational Conference on Management of Data (SIGMOD ’06), Pages: 229-240, in the year 2006.

[18] KVM, http://www.linux-kvm.org/page/Main_Page, 2013.

[19] OpenStack, http://openstack.org/, 2013

[20] Apache, “Hadoop,”http://hadoop.apache.org, 2013.

[21] UCI Machine Learning Repository, ftp://ftp.ics.uci.edu/pub/ machine-learning-databases/, 2013.