ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization using MapReduce on Cloud

Ranjana Nadagoudar[1], Radhika[2]

Associate Professor, Dept. of Computer Science and Engineering, VTU RO Centre, Kalaburgi, India[1]

IV Sem M. Tech, Dept. of Computer Science and Engineering, VTU RO Centre, Kalaburgi, India[2]

Abstract: A large number of cloud forces require users to carve up private data like electronic health records for data analysis or mining, bringing privacy concerns. Anonymizing data sets via generalization to satisfy certain privacy requirements such as k-anonymity is a widely used category of privacy preserving techniques. At present, the scale of data in many cloud applications increases massively in accordance with the Big Data trend, thereby making it a challenge for commonly used software tools to confine, manage, and process such large-scale data within a adequate elapsed time. As a result, it is a challenge for existing anonymization approaches to accomplish privacy preservation on privacy-sensitive large-scale data sets due to their insufficiency of scalability. In this paper, we propose a scalable two phase top-down specialization (TDS) to anonymize large-scale data sets using the MapReduce framework on cloud. In both phases of our approach, we deliberately design a group of inventive MapReduce jobs to concretely accomplish the specialization computation in a highly scalable way. Experimental assessment results demonstrate that with our approach, the scalability and efficiency of TDS can be significantly enhanced over existing approaches.

Keywords: Data anonymization, top-down specialization, MapReduce, cloud, privacy preservation

I. INTRODUCTION

CLOUD computing, a disruptive trend at present, poses a considerable impact on current IT industry and research communities [1]. Cloud computing provides massive computation power and storage capacity via utilizing a large number of commodity computers together, enabling users to organize applications cost-effectively without heavy infrastructure asset. Cloud users can reduce huge upfront investment of IT infrastructure, and focus on their own core business. However, numerous possible customers are still diffident to take advantage of cloud due to privacy and security concerns [5]. The research on cloud privacy and security has come to the depiction [9]. Privacy is one of the most concerned issues in cloud computing, and the concern aggravates in the perspective of cloud computing although some privacy issues are not new[1], [5]. Personal data like electronic health records and financial transaction records are usually deemed extremely sensitive although these data can present significant human benefits if they are analyzed and mined by organizations such as disease research centres. Data privacy can be divulged with less effort by malevolent cloud users or providers because of the failures of some conventional privacy protection measures on cloud [5]. This can bring substantial economic loss or strict social reputation mutilation to data owners. Hence, data privacy issues need to be addressed urgently before data sets are analyzed or pooled on cloud. Data anonymization has been extensively studied and widely adopted for data privacy preservation in non interactive data publishing and sharing scenarios [11]. Data anonymization refers to hiding characteristics and/or sensitive data for owners of data records. Then, the privacy of an individual can be effectively preserved while certain increasing information is exposed to data users for various analysis and mining. A variety of anonymization algorithms with different anonymization operations have been proposed [15]. However, the scale of data sets that need anonymizing in some cloud applications increases tremendously in accordance with the cloud computing and Big Data trends [1]. Data sets have become so large that anonymizing such data sets is becoming a considerable challenge for conventional anonymization algorithms. The researchers have begin to investigate the scalability problem of large-scale data anonymization. Large-scale data processing frameworks like MapReduce have been included with cloud to provide dominant 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 analogous 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 trade off between data effectiveness and data reliability, is widely applied for data anonymization [12]. Most TDS algorithms are centralized, resulting in their insufficiency in handling large scale data sets. Although some dispersed algorithms have been proposed they mainly focus on secure have been proposed they mainly focus on secure anonymization of data sets from numerous parties, rather than the scalability aspect. As the MapReduce computation hypothesis is moderately 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

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

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 data sets. We leverage MapReduce to accomplish the concrete computation in both phases. A group of MapReduce jobs is purposely designed and corresponding to perform specializations on data sets collaboratively.

The major contributions of our research are threefold. First, we cratively apply MapReduce on cloud to TDS for data anonymization and consciously design a group of innovative MapReduce jobs to concretely accomplish thespecializations in a highly scalable fashion. Second, we propose a two-phase TDS approach to gain high scalability via allowing specializations to be conducted on several data partitions in parallel during the first phase. Third, experimental results show that our approach can significantly improve the scalability and efficiency of TDS for data anonymization over existing approaches. The remainder of this paper is organized as follows: The next section reviews related work, and analyzes the scalability problem in existing TDS algorithms.

The remainder of this paper is organized as follows: The next section reviews related work, and analyzes the scalability problem in existing TDS algorithms. In Section III, we briefly present proposed system for our approach. Section IV preliminary approach, and Section 5 formulates the two-phase TDS approach. In Section 6 formulates mapreduce version of centralized TDS. Finally, we conclude this paper.

II. . RELATED WORK AND PROBLEM ANALYSIS

A. Related Work

Recently, data privacy preservation has been extensively investigated [11]. We briefly review related work below. LeFevre et al. [17] addressed the scalability problem of anonymization algorithms via introducing scalable decision trees and sampling techniques. Iwuchukwu and Naughton [18] proposed an R-tree index-based approach by building a spatial index over data sets, achieving high efficiency. However, the above approaches aim at multidimensional generalization [15], thereby failing to work in the TDS approach. Fung et al. [12], [20], [21] proposed the TDS approach that produces anonymous data sets without the data exploration problem [11]. A data structure Taxonomy Indexed PartitionS (TIPS) is subjugated to improve the efficiency of TDS. But the approach is centralized, leading to its insufficiency in handling large-scale data sets. Several distributed algorithms are proposed to preserve privacy of multiple data sets retained by multiple parties. Jiang and Clifton [24] and Mohammed et al. [22] proposed distributed algorithms to anonymize vertically partitioned data from different data sources without disclosing privacy information from one party to another. Jurczyk and Xiong [25] and Mohammed et al. [20] proposed distributed algorithms to anonymize horizontally partitioned data sets retained by multiple holders. However, the above distributed algorithms mainly aim at securely integrating and anonymizing multiple data sources. Our research mainly focuses on the scalability issue of TDS anonymization, and is, therefore, orthogonal and complementary to them. As to MapReduce-relevant privacy protection, Roy et al. [26] investigated the data privacy problem caused by MapReduce and presented a system named Airavat incorporating mandatory access control with differential privacy. Further, Zhang et al. [27] leveraged MapReduce to automatically partition a computing job in terms of data security levels, protecting data privacy in hybrid cloud. Our research exploits MapRedue itself to anonymize large-scale data sets before data are further processed by other MapReduce jobs, arriving at privacy preservation.

B.Problem Analysis

We analyze the scalability problem of existing TDS approaches when handling large-scale data sets on cloud. The centralized TDS approaches in [12], [20], and [21] exploits the data structure TIPS to improve the scalability and efficiency by indexing anonymous data records and retaining statistical information in TIPS. The data structure speeds up the specialization process because indexing structure avoids frequently scanning entire data sets and storing statistical results circumvents recomputation overheads. On the other hand, the amount of metadata retained to maintain the statistical information and linkage information of record partitions is relatively large compared with data sets themselves, thereby consuming considerable memory. Moreover, the overheads incurred by maintaining the linkage structure and updating the statistic information will be huge when date sets become large. Hence, centralized approaches probably suffer from low efficiency and scalability when handling large-scale data sets. There is an assumption that all data processed should fit in memory for the centralized approaches [12]. Unfortunately, this assumption often fails to hold in most data-intensive cloud applications nowadays. In cloud environments, computation is provisioned in the form of virtual machines (VMs). Usually, cloud compute services offer several flavors of VMs. As a result, the centralized approaches are difficult in handling large-scale data sets well on cloud using just one single VM even if the VM has the highest computation and storage capability. A distributed TDS approach [20] is proposed to address the distributed anonymization problem which mainly concerns privacy protection against other parties, rather than scalability issues. Further, the approach only employs information gain, rather than its combination with privacy loss, as the search metric when determining the best specializations. As pointed out in [12], a TDS algorithm without considering privacy loss probably chooses a specialization that leads to a quick violation of anonymity requirements. Hence, the distributed algorithm fails to produce anonymous data sets exposing the same data

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

utility as centralized ones. Besides, the issues like communication protocols and fault tolerance must be kept in mind when designing such distributed algorithms. As such, it is inappropriate to leverage existing distributed algorithms to solve the scalability problem of TDS.

III.PROPOSED SYSTEM

In this paper, we propose a scalable two-phase top-down specialization (TDS) approach to anonymize large-scale data sets using the MapReduce framework on cloud. In both phases of our approach, we deliberately design a group of innovative MapReduce jobs to concretely accomplish the specialization computation in a highly scalable way. This approach get input data's and split into the small data sets. Then we apply the ANONYMIZATION on small data sets to get intermediate result. Then small data sets are merge and again apply the ANONYMIZATION. We analyze the each and every data set sensitive field and give priority for this sensitive field. Then we apply ANONYMIZATION on this sensitive field only depending upon the scheduling.

A.ADVANTAGES OF PROPOSED SYSTEM

- Accomplish the specializations in a highly scalable fashion.
- Gain high scalability.
- Significantly improve the scalability and efficiency of TDS for data anonymization over existing approaches.
- The overall performance of the providing privacy is high.
- Its ability to handles the large amount of dat sets.
- The anonymization is effective to provide the privacy on data sets.
- Here we using the scheduling strategies to handle the high amount of datasets.

IV.PRELIMINARY

A.Basic Notations

We describe several basic notations for convenience. Let D denote a data set containing data records. A record $r \in D$ has the form $r = (v1, v2, \ldots, vm, sv)$, where m is the number of attributes, $vi, 1 \leq i \leq m$, is an attribute value and *sv* is a sensitive value like diagnosis. The set of sensitive values is denoted as SV. An attribute of a record is denoted as *Attr*, and the taxonomy tree of this attribute is denoted as TT. Let DOM represent the set of all domain values in TT. The quasi-identifier of a record is denoted as qid = hq1, q2, . . . ,qmi, where qi € DOMi. Quasi identifiers, representing groups of anonymous records, can lead to privacy breach if they are too specific that only a small group of people are linked to them [11]. Quasi-identifier set is denoted as QID =(Attr1, Attr2, . . . , Attrm). The set of the records with qid is defined as QI-group [28], denoted by QIG(qid). QI is the acronym of quasi-identifier.Without loss of generality, we adopt k-anonymity [23] as the privacy model herein, i.e., for any qid € QID, the size of G(qid) must be zero or at least k. Otherwise, the individuals owning such a quasi-identifier can be linked to sensitive information with higher confidence than expected, resulting in privacy breach. The k-anonymity privacy model can combat such a privacy breach because it ensures that an individual will not be distinguished from other at least k _ 1 ones. The anonymity parameter k is specified by users according to their privacy requirements. In the TDS approach, a data set is anonymized via performing specialization operations. A specialization operation is to replace a domain value with all its child values. Formally, a specialization operation is represented as spec : p → Child(p), where p is a domain value and Child(p) DOM is the set of all the child values of p. The domain values of an attribute form a "cut" through its taxonomy tree [11]. The cut of attribute Attri, denoted as Cuti, $1 \leq i \leq m$, is a subset of values in DOMi . Cuti contains exactly one value in each root-to-leaf path in taxonomy tree TTi. The cuts of all attributes determine the anonymity of a data set. To capture the degree of anonymization intuitively during the specialization process, we give the subsequent definition.

Definition 1. (Anonymization Level). A vector of cuts of all attributes is defined as anonymization level, denoted as AL. Formally, AL = (Cut1; Cut2; . . . ; Cutm), where Cuti, $1 \leq i \leq m$ is the cut of taxonomy tree TTi.

Anonymization level can intuitively represent the anonymization degree of a data set, i.e., the more specific AL a data set has, the less degree of anonymization it corresponds to. Thus, TDS approaches employ anonymization level to track and manage the specialization process.

B.Top-Down Specialization

Generally, TDS is an iterative process starting from the topmost domain values in the taxonomy trees of attributes. Each round of iteration consists of three main steps, namely, finding the best specialization, performing specialization and updating values of the search metric for the next round [12]. Such a process is repeated until k-anonymity is violated, to expose the maximum data utility. The goodness of a specialization is measured by a search metric. We adopt the information gain per privacy loss (IGPL), a trade off metric that considers both the privacy and information requirements, as the search metric in our approach. A specialization with the highest IGPL value is regarded as the best one and selected in each round.
Given a specialization spec : p → Child(p), the IGPL of the specialization is calculated by

$$IGPL(spec) = IG(spec)=(PL(spec) + 1). \quad (1)$$

The term IG(spec) is the information gain after performing spec, and PL(spec) is the privacy loss. IG(spec) and PL(spec) can be computed via statistical information derived from data sets.

V. TWO-PHASE TOP DOWN SPECIALIZATION(TPTDS)

A.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

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

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 [29]. 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 anonymize entire data sets. Details are formulated as follows. In the first phase, an original data set D is partitioned into smaller ones.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. MRTDS anonymizes data partitions to generate intermediate anonymization levels. An intermediate anonymization level means that further specialization can be performed without violating k anonymity. MRTDS only leverages the task level parallelization of MapReduce. Formally, let function MRTDS(D, k, AL) → AL0 represent a MRTDS routine that anonymizes data set D to satisfy k-anonymity from anonymization level AL to AL0. Thus, a series of functions MRTDS(Di, kI, AL0) → AL0i, 1 ≤ i ≤ p, are executed simultaneously in the first phase. The term kI denotes the intermediate anonymity parameter, usually given by application domain experts. Note that kI should satisfy kI ≥ k to ensure privacy preservation. AL0 is the initial anonymization level, i.e., AL0 = ({Top1}, {Top2}, . . ., {Topm}), where Topj € DOMj, 1 ≤ j ≤ m, is the topmost domain value in TTi. AL0 i is the resultant intermediate anonymization level. In the second phase, 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((AL01,AL02, . . .;AL0p)) → ALI .Then, the whole data set D is further anonymized based on ALI , achieving k-anonymity finally, i.e., MRTDS(D, k, ALI) → AL*, where AL* denotes the final anonymization level.

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, kI, AL0) → AL0i, 1 ≤ i ≤ p in parallel as multiple MapReduce jobs.
3: Merge all intermediate anonymization levels into one, Merge(AL01, AL02, . . ., AL0p) → ALI .
4: Execute MRTDS(D, k, ALI) → AL* to achieve k-anonymity.
5: Specialize D according to AL*, Output D*.

In essential, TPTDS divides specialization operations required for anonymization into the two phases. Let SP1i, 1 ≤ i ≤ p, denote the specialization sequence on Di in the first phase, i.e., SP1i = (speci1, speci2, . . . ; speciji), where ji is the number of specializations. The first common subsequence of SP1i, 1 ≤ i ≤ p, is denoted as SPI . Let SP2 denote the specialization sequence in the second phase. SP2 is determined by ALI rather than kI . Specifically, more specific ALI implies smaller SP2. Throughout TPTDS, the specializations in the set SPI ∪ SP2 come into effect for anonymization. The influence of p and kI on the efficiency is analyzed as follows. Greater p means higher degree of parallelization in the first phase, and less kI indicates more computation is conducted in the first phase. Thus, greater p and less kI can improve the efficiency.

B.Data Partition

When D is partitioned into Di, 1 ≤ i ≤ p, it is required that the distribution of data records in Di is similar to D.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 from Di, 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 random sample of D.

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, we run MRTDS(Di, kI, AL0) on these data sets in parallel to derive intermediate anonymization levels *ALi , 1 ≤ i ≤ p.*

C. Anonymization Level Merging

All intermediate anonymization levels are merged into one in the second phase. The merging of anonymization levels is completed by merging cuts. Specifically, let Cuta in AL0a and Cutb in AL0b be two cuts of an attribute. There exist domain values qa € Cuta and qb € Cutb that satisfy one of the three conditions: qa is identical to qb, qa is more general than qb, or qa is more specific than qb. To ensure that the merged intermediate anonymization level ALI never violates privacy requirements, the more general one

DOI 10.17148/IJARCCE.2015.45116

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

is selected as the merged one, for example, qa will be selected if qa is more general than or identical to qb. For the case of multiple anonymization levels, we can merge them in the same way iteratively. The following lemma ensures that ALI still complies privacy requirements.

D.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, ALI) 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 Algorithm3. 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, . . . , pm, sv), pi, 1 ≤ i ≤ m, is the parent of a specialization in current AL and is also an ancestor of vi in r, emit (r*,count).
Reduce: For each r*, sum ← ∑count, emit (r*, sum).

VI. MAPREDUCE VERSION OF CENTRALIZED TDS

We elaborate the MRTDS in this section. MRTDS plays a core role in the two-phase TDS approach, as it is invoked in both phases to concretely conduct computation. Basically, a practical MapReduce program consists of Map and Reduce functions, and a Driver that coordinates the macro execution of jobs.

A. MRTDS Driver

Usually, a single MapReduce job is inadequate to accomplish a complex task in many applications. Thus, a group of MapReduce jobs are orchestrated in a driver program to achieve such an objective. MRTDS consists of MRTDS Driver and two types of jobs, i.e., IGPL Initialization and IGPL Update. The driver arranges the execution of jobs.

B. IGPL Initialization Job

The main task of IGPL Initialization is to initialize information gain and privacy loss of all specializations in the initial anonymization level AL.

C.IGPL Update Job

The IGPL Update job dominates the scalability and efficiency of MRTDS, since it is executed iteratively. So far, iterative MapReduce jobs have not been well supported by standard MapReduce framework like Hadoop. The IGPL Update job is quite similar to IGPL Initialization, except that it requires less computation and consumes less network bandwidth.

Since the IGPL Update job dominates the scalability and efficiency of MRTDS, we briefly analyze its complexity as follows. Let n denote all the records in a data set, m be the number of attributes, s be the number of mappers, and t be the number of reducers. As a mapper emits (m + 1) key-value pairs, it takes O(1) space and O(m + n/s) time.Similarly, a reducer takes O(1) space and O(m + n/t) time. Note that a reducer only needs O(1) space due to the MapReduce feature that the key-value pairs are sorted in the shuffle phase. Otherwise, the reducer needs more space to accumulate statistic information for a variety of specializations. The communication cost is O(m + n) according to the map function, but communication traffics can be reduced heavily by optimization techniques like Combiner.

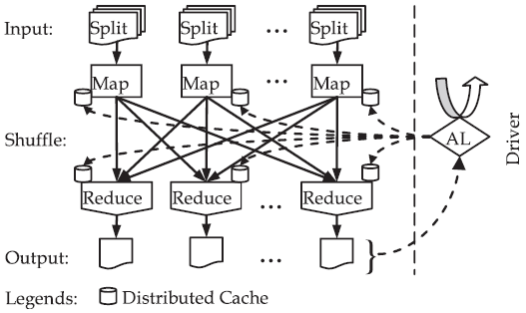

Fig.1. Execution framework overview of MRTDS.

D.Implementation and Optimization

To elaborate how data sets are processed in MRTDS, the execution framework based on standard MapReduce is depicted in Fig. 1. The solid arrow lines represent the data flows in the canonical MapReduce framework. From Fig. 1, we can see that the iteration of MapReduce jobs is controlled by anonymization level AL in Driver. The data flows for handling iterations are denoted by dashed arrow lines. AL is dispatched from Driver to all workers including Mappers and Reducers via the distributed cache mechanism. The value of AL is modified in Driver according to the output of the IGPL Initialization or IGPL Update jobs. As the amount of such data is extremely small compared with data sets that will be anonymized, they can be efficiently transmitted between Driver and workers. We adopt Hadoop, an open-source implementation of MapReduce, to implement MRTDS. Since most of Map and Reduce functions need to access current anonymization level AL, we use the distributed cache mechanism to pass the content of AL to each Mapper or Reducer node as shown in Fig. 1. Also, Hadoop provides the mechanism to set simple global variables for Mappers and Reducers. The best specialization is passed into the Map function of IGPL Update job in this way. The partition hash function in the shuffle phase is modified because the two jobs require that the key-value pairs with the same key:p field rather than entire key should go to the same Reducer. To reduce communication traffics, MRTDS exploits combiner mechanism that aggregates the key-value pairs with the same key into one on the nodes running Map functions.

DOI 10.17148/IJARCCE.2015.45116

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

VI. CONCLUSION

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 intentionally designed a group of inventive MapReduce jobs to concretely achieve the specialization computation in a highly scalable way. The scalability and efficiency of TDS are improved significantly over existing approaches.

ACKNOWLEDGMENT

I would like to express my sence of proud gratitude and indebtedness to my guide, for his valuable guidance, suggestions, timely supervision for successful completion of my paper. Above all I would like to thanks all my family members and friends for their constructive critism and construct support in making this grand success.

REFERENCE

[1] S. Chaudhuri, "What Next?: A Half-Dozen Data Management Research Goals for Big Data and the Cloud," Proc. 31st Symp. Principles of Database Systems (PODS '12), pp. 1-4, 2012.

[2] M. Armbrust, A. Fox, R. Griffith,, Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, 2010.

[3] L. Wang, J. Zhan, W. Shi, and Y. Liang, "In Cloud, Can Scientific Communities Benefit from the Economies of Scale?," IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 2, pp.296-303, Feb. 2012.

[4] H. Takabi, J.B.D. Joshi, and G. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," IEEE Security and Privacy, vol. 8, no. 6, pp. 24-31, Nov. 2010.

[5] D. Zissis and D. Lekkas, "Addressing Cloud Computing Security Issues," Future Generation Computer Systems, vol. 28, no. 3, pp. 583- 592, 2011.

[6] X. Zhang, C. Liu, S. Nepal, S. Pandey, and J. Chen, "A Privacy Leakage Upper-Bound Constraint Based Approach for Cost-Effective Privacy Preserving of Intermediate Data Sets in Cloud," IEEE Trans. Parallel and Distributed Systems, to be published, 2012.

[7] L. Hsiao-Ying and W.G. Tzeng, "A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding," IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 6, pp. 995- 1003, 2012.

[8] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," Proc. IEEE INFOCOM, pp. 829-837, 2011.

[9] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler, "Gupt: Privacy Preserving Data Analysis Made Easy," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '12), pp. 349- 360, 2012.

[10] Microsoft HealthVault, http://www.microsoft.com/health/ww/products/Pages/healthvault.aspx, 2013.

[11] B.C.M. Fung, K. Wang, R. Chen, and P.S. Yu, "Privacy-Preserving Data Publishing: A Survey of Recent Devel- opments," ACM Computing Surveys, vol. 42, no. 4, pp. 1-53, 2010.

[12] B.C.M. Fung, K. Wang, and P.S. Yu, "Anonymizing Classification Data for Privacy Preservation," IEEE Trans. Knowledge and Data Eng., vol. 19, no. 5, pp. 711-725, May 2007.

[13] X. Xiao and Y. Tao, "Anatomy: Simple and Effective Privacy Preservation," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB'06), pp. 139-150, 2006.

[14] 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), pp. 49-60, 2005.

[15] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, "MondrianMultidimensional K-Anonymity," Proc. 22nd Int'l Conf. Data Eng. (ICDE '06), 2006.

[16] 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), pp. 3-14, 2012.

[17] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, "Workload-Aware Anonymization Techniques for Large-Scale Data Sets," ACM Trans. Database Systems, vol. 33, no. 3, pp. 1-47, 2008.

[18] T. Iwuchukwu and J.F. Naughton, "K-Anonymization as Spatial Indexing: Toward Scalable and Incremental Anonymization," Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB '07), pp.746- 757,2007.

[19] J. Dean and S. Ghemawat, "Mapreduce: Simplified Data rocessing on Large Clusters," Comm. ACM, vol. 51, no. 1, pp. 107-113, 2008.

[20] N. Mohammed, B. Fung, P.C.K. Hung, and C.K. Lee, "Centralized and Distributed Anonymization for High-Dimensional Healthcare Data," ACM Trans. Knowledge Discovery from Data, vol. 4, no.4 Article 18, 2010.

[21] B. Fung, K. Wang, L. Wang, and P.C.K. Hung, "Privacy-Preserving Data Publishing for Cluster Analysis," Data and Knowledge Eng., vol. 68, no. 6, pp. 552-575, 2009.

[22] N. Mohammed, B.C. Fung, and M. Debbabi, "Anonymity Meets Game Theory: Secure Data Integration with MaliciousParticipants," VLDB J., vol. 20, no. 4, pp. 567-588, 2011.

[23] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," Int'l J. Uncertainty, Fuzziness and Knowledge-Based Systems,vol. 10, no. 5, pp. 557-570, 2002.

[24] W. Jiang and C. Clifton, "A Secure Distributed Framework for Achieving k-Anonymity," VLDB J., vol. 15, no. 4, pp. 316-333, 2006