Indian Journal of Science and Technology, *vol 8(22), DOI: 10.17485/ijst/2015/v8i22/79095, September 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Parallel Bottom-up Generalization Approach for Data Anonymization using Map Reduce for Security of Data in Public Cloud

Amalraj Irudayasamy* **and L. Arockiam**

[1]Periyar University, Salem - 636011, Tamil Nadu, India; amalprisci@gmail.com
[2]Computer Science, St. Joseph's College, Trichy - 620002, Tamil Nadu, India; larockiam@yahoo.co.in

Abstract

BackGround/Objectives: Anonymizing data sets through generalization satisfies certain privacy concerns such as k-anonymity that are broadly used as privacy conserving procedures. Parallel bottom-up generalization approach is introduced to anonymize huge datasets by map reduce structure on public cloud. A group of innovative map reduce jobs are formulated to perform the generalization in an exceedingly scalable manner. **Methods/Statistical Analysis:** Map Reduce, a widely-adopted parallel data processing framework is introduced, to address the privacy preservation problem with minimum information loss of the Bottom-Up Generalization (BUG) approach for large-scale data anonymization. To make full use of parallelism feature of Map reduce on cloud the whole process are split into two phases. Firstly, unique datasets are partitioned into a collection of lesser datasets, and these datasets are anonymized in parallel, giving intermediary outcome. Secondly, the intermediate results are combined and anonymized, to attain consistent *k*-anonymous data sets. Map Reduce concept is used to accomplish the computation in both phases. **Findings:** In this paper, investigational evaluation, results to gain high privacy preservation with minimum information loss in less execution time when compared to the existing approaches. The results demonstrate the insufficiency of the state-of-the art sub-tree anonymization approaches when handling large data sets. According to the tendencies of execution time and Information Loss, it is necessary and reasonable to choose MRBUG to perform parallel generalized data anonymization for large data according to the value of *k*. **Applications/Improvements:** Optimized, heuristic, and balanced scheduling approaches are expected to be developed towards overall scalable privacy preservation. It is believed that the structure of bottom-up generalization is amenable to several extensions that make it more practical. Incorporating different metrics and handling data suppressions in partial generalization is not necessarily require to have all child values generalized altogether. It is also possible to generalize numeric attributes without a pre-determined hierarchy and shall be taken up as a future work.

Keywords: Bottom-Up Generalization, Cloud, Data Anonymization, Map Reduce, Privacy Preservation

1. Introduction

Cloud computing, a new development gives a significant influence on current IT industry and research fraternity.[1-3] Cloud computing offers huge computation facilities and storage capacity via using many computers called clusters together, provide users to position applications cost effectively without heavy infrastructure outlay. Cloud users may decrease huge cost of investment and focus on their own core business. However many organizations are still reluctant to make use of the advantages of cloud computing due to privacy and security concerns[4,5]. The study on cloud privacy and security has taken its full swing[6-9]. Privacy is the major issue in cloud computing[1,5]. Data privacy may be exposed with less effort by hackers because of the failures of some traditional security measures[5]. This may bring substantial loss economically or social impairment to data owners. Hence, data privacy issues need to be enhanced before data sets are published on public cloud. Data anonymization has been comprehensively considered and broadly adopted for privacy preservation[10,11].

*Author for correspondence

Data anonymization refers to hiding the identity of sensitive data. The privacy of an individual may be effectively preserved with certain aggregate information exposed to users for diverse analysis. A variety of anonymization algorithms with various operations have been proposed[12-15] Data sets have become so large that anonymizing such data sets are becoming a considerable challenge for traditional algorithms[1,16]. The researchers have begun to investigate the scalability problem of large-scale data anonymization[17,18]. Large-scale data processing frameworks like Map reduce[19] 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 this research, Map reduce, a widely-adopted parallel data processing framework is introduced, to address the scalability problem of the Bottom-Up Generalization (BUG) approach[12] for large-scale data anonymization. The BUG approach, offers a good tradeoff between data utility and data consistency, is widely applied for data anonymization[20-22]. Most BUG algorithms are centralized, resulting in their inadequacy in handling large-scale data sets. Although some distributed algorithms have been proposed, each one mainly focuses on secure anonymization of data sets from multiple parties, than the scalability aspect[23].

In this paper, an efficient parallel BUG approach for Data anonymization is proposed using Map reduce. To evaluate this approach experiments on real-world data sets are conducted. Experimental results demonstrate that with this approach, the scalability and efficiency of BUG may be improved significantly over existing approaches. The major contributions of this research are threefold. First, creatively applying Map reduce on cloud to BUG for data anonymization and formulate a group of innovative Map reduce jobs to accomplish the generalizations in a highly scalable fashion. Second, a two-phase BUG approach to gain high scalability via allowing generalizations to be conducted on multiple data partitions in parallel during the first phase is proposed. Third, an experimental results show that this approach may significantly improve the scalability and efficiency of BUG for data anonymization over existing approaches.

The remainder of this paper is organized as follows. The next section reviews background work, and analyzes the scalability problem in existing BUG algorithms. In Section three, preliminaries for BUG are briefly presented. Section four formulates the parallel BUG approach, and Section five elaborates algorithmic details of Map reduce jobs. Empirical evaluation has been made for the proposed approach in Section six. Finally, the conclusion and discussion over the future work has been delivered in Section 6.

2. Background and Problem Investigation

A well-studied method for hiding sensitive information with statistical methods is randomizing sensitive attributes by totaling random error to values[24]**.** In these mechanism, privacy was quantified by the unique values of a randomized characteristic may be assessed. This method is dissimilar to the *k*-anonymity that quantifies the individual that may be connected to an outer source. A systematic revision is carried out in data mining for masking data. Preferences may be carried through the taxonomical hierarchies and the data receiver may be articulated the change to data so that the product may be accurately inferred. Generalization was used to attain anonymity in Data fly and μ-Argus systems[25]**.** Their mechanisms did not classify or specific the use of unconfined data. Data falsification is measured by several hierarchy levels[26] Selection of attributes did not address the quality for classification[26]**.** Generalization approach considers the anonymity problem for classification, and presented an algorithm to search the best generalization of the data[27]**.** It is more time consuming to generalize a small quantity of records. The iterative bottom-up generalization is used, and concentrates on the scalability concern. An information privacy protocol has been used to generalize, whereas researches have been carried out to filter a selected generalization[28].

Many distributed procedures are projected to preserve privacy of multiple data sets. Jiang et al.[24] and Mohammed et al.[22] proposed disseminated procedures to anonymize vertically partitioned data from diverse data sources without disclosing privacy information from one party to a different one. Jurczyk et al.[28] and Mohammed et al.[20] proposed distributed algorithms to anonymize horizontally partitioned data sets retained by multiple holders. However, the above mentioned distributed procedures mostly intend at securely integrating and anonymizing multiple data sources. This research mainly focuses on the privacy preservation problem achieving minimum information loss in less execution time. Roy et al.[29] studied the data privacy problem caused by Map Reduce

and presented "Airavat" incorporating compulsory right to use control with discrepancy privacy. Further, Zhang et al.[30] projected Map Reduce to repeatedly partition a computing job for data security levels, protecting data privacy in hybrid cloud. This research investigates Map Reduce to anonymize large-scale data sets, attaining maximum privacy preservation.

3. Preliminary

3.1 Basic Notations

Consider that the owner wants to release a person private data $R(D_1,\ldots\ldots,D_n,C)$ to the cloud. A record has the form $<v_1,\ldots..,v_n,cls>$, where v_i is a domain value of the attribute D_i and cls is a class in C. Assume that R shares certain attributes with an outside source E, represents $R \cap E$. If a value on $R \cap E$ is so specific, if the probability of having this value is trivial, each link from a record in R to certain information in E has a good chance of recognizing real life information. The owner safeguards the data against such linkages by using a least amount of records that is linked over each value on $R \cap E$. Let DOM represents the set of all domain values in the Taxonomy Encoded Anonymity (TEA).

3.1.1 Definition 1 (Anonymity)

The Virtual Identifier, Denoted VID, is the set of attributes combined by R and E. a(vid) represents the number of records in R with the value vid on VID. The anonymity of VID, denoted as A (VID), is the minimum a(vid) for several value vid on VID. If a(vid) = A(VID), vid is known an anonymity vid.

3.1.2 Definition 2 (Generalization)

A generalization, written $\{c\} \rightarrow p$ replaces all child values $\{c\}$ along with the parental value p. A generalization is efficient if all values under c are generalized to c. A vid is generalized $\{c\} \rightarrow p$ if the vid contains some value in $\{c\}$.

3.2 Metrics for Generalization

A superior generalization ought to protect privacy of the information and focus to achieve the *K*-anonymity. Assume a generalization $G{:}\{c\} \rightarrow p$. Let *Rc denotes* the set of records with *c*, and let *Rp* denote the set of records with p after applying $G.|R_p| = \Sigma_c|R_c|$, where $|x|$ is the number of elements in a bag *x*. The result of G is reported by the information loss and anonymity gain after substituting R_c's with R_p. There are two metrics namely Entropy Based Information Loss (EBIL) and Information-Privacy Metric to reduce the information loss thereby achieving K-anonymity.

3.3 Bottom-up Generalization

A bottom-up approach is organizing smaller systems together which results to more complex systems. Bottom-up processing is a type of information processing based on incoming data from the environment to form a perception. In a bottom-up approach the discrete components of the system are initially specified in pronounced detail. These components are then interrelated to form greater subsystems, occasionally in many stages, till a comprehensive top-level system is shaped. This strategy often resembles a seed classic, whereby the openings are small but ultimately develop in complication and completeness.

A generalization operation is to replace a value with its parent in a taxonomy tree, while a specialization operation is to replace a value with its all child values. Formally, a generalization is represented as *gen: child(q)* $\rightarrow q$, where q is a domain value and the set *Child(q)* consists of all child domain values of q. The domain values of a taxonomy tree in any stage of anonymization comprise a cut through the tree. Formally, the cut of the taxonomy tree of the attribute $Attr_i$ denoted as $Cuti1 \leq i \leq m$, is a subset of values in *DOMi*. *Cuti* contains exactly one value in each root-to-leaf path in the taxonomy tree *TTi*. Intuitively, the cuts of all attributes determine the anonymity of an anonymous data set. Hence, such information is leveraged to capture the degree of anonymization during anonymization process.

Anonymization Level (AL) can intuitively represent the anonymization degree of an anonymous data set, i.e., the more specific AL a data set has, less anonymity it corresponds to. In fact, anonymization can be achieved by generalizing or specializing AL. After obtaining an AL, the original data can be recoded according to the AL to produce the final anonymous data. AL is employed to track and manage an anonymization process.

In BUG, the Information Loss per Privacy Gain (ILPG) is leveraged as the search metric for BUG. Given generalization *gen: child (q)* $\rightarrow q$, the ILPG of the generalization is calculated. The term *IL(gen)* is the information loss after performing gen, and *PG(gen)* is the privacy gain. Both of them are computed via

statistical information derived from data sets and *IL(gen)* is calculated. Let *Aq(gen)* denote the anonymity after performing gen, while *Ac(gen)* be that before performing gen and the privacy gain from gen is calculated.

4. Map Reduce BUG

Map Reduce is mainly elaborated based Bottom-Up Generalization (MRBUG) in this section. MRBUG Driver is described in section 4.1 to present the basic process of BUG. To improve the competence of this approach, the parallelization degree of BUG is boosted in section 4.2. Section 4.3 presents the Map Reduce job for computing IGPL in detail.

4.1 MRBUG Driver

Principally, BUG is a repeated process opening from the lowest Anonymization Level (AL). The lowest AL has the inner domain nodes in the lowest level of taxonomy trees. Each round of iteration includes four steps, namely, checking the current data set satisfies the anonymity requirement, manipulating the Information Loss per Privacy Gain (ILPG), finding the best generalization and generalizing the data set according to the selected best generalization.

Procedure 4.1 presents the Map Reduce driver for MRBUG. ILPG values of all generalizations are initialized in Step two. Step three is the main procedure, which is an iterative process. It checks whether the current anonymized data set satisfies k-anonymity. If yes, it proceeds to Step 4. Otherwise, a serial of actions are triggered.

4.1.1 Procedure: MRBUG Driver

- Input data set D , the lowest anonymization level AL0 and k-anonymity parameter k.
- Initialize the values of search metric ILPG for each generalization with respect to AL0, through job ILPG Calculation.
- When generalization is below the anonymity parameter k , then do the following :
  - Identify the available generalization set AG Set out of all the active generalization candidates.
  - Set generalization as inactive for all generalization that belongs to AG Set , to perform generalization on the current anonymization level.
  - If the generalization is inactive then do the following:
    - Insert a new generalization into NG Set.
    - Remove all generalization in .
  - Update ILPG values for all active generalization candidates, through ILPG calculation.
- Anonymize D to D* in terms of resultant anonymization level through data anonymization job.
- Anonymous data set D* is got as output.

Step 3, 1 identifies the available generalization set AG. Initially, *AGSet* only contains the best generalization *gen Best* with the highest ILPG value in terms of the conventional BUG process. But it is proposed to perform multiple generalizations in one round of iteration in MRBUG to improve the degree of parallelization and efficiency, which will be elaborated in Section 3.4.2. Step 3, 2 performs the generalizations in *AGSet* by labelling them as INACTIVE. If a generalization is labelled as INACTIVE, it will not be considered any more in following rounds, abstractly fulfilling anonymization on the data set. Let *SGS*(gen) denote the set containing generalization gen and it's all siblings in the domain taxonomy tree. When the generalizations in **SGSet**(**gen**) are all labeled as INACTIVE, a new higher level generalization is inserted into the AL to replace these inactive ones, as specified in Step 3, 3. Note that this is a remarkable difference from TDS. Since multiple generalizations in AGSet are checked for this, it is probably that more than one new generalization produced. Let NGSet be the set of such generalizations. So, Step 3, 3, 1 adds new generalizations to *NGSet*. Step 3, 4 updates the privacy gain of each active generalization as performing of generalizations in *AGSet* probably changes the anonymity of the data set. Also, information loss computation is required if new generalizations have been inserted.

As the last step, Step 4 concretely anonymizes the data set according to the final AL. Step 2 and Step 3.4 requires ILPG calculation that involves accessing to the original data set and computing statistic information over the data set. Map Reduce technique has been used to conduct the intensive computation in these situations. Specifically, an innovative Map Reduce job ILPG Calculation is designed to accomplish the computation required in Step 2 and Step 3.4. The job is elaborated in Section 3.4.3. In the following section, we discuss how to boost parallelization of performing generalization in a round of iteration to improve scalability and efficiency of BUG.

4.2 Parallelization of Performing Generalization

Several observations probably help to design efficient Map Reduce jobs for the ILPG calculation. One is that, unlike TDS that inserts several new specialization candidates into the current anonymization level in each round, BUG only inserts a new generalization candidate after several rounds of generalization. Another is that conducting a generalization will not affect the information loss of another generalization candidate. Based on such observations, multiple generalization candidates can be taken into account in one round, thereby improving the degree of parallelization and the efficiency of proposed approach. However, performing a generalization possibly changes the anonymity of the data set and privacy gain of each candidate will be affected. The following definitions identify which candidates can be considered simultaneously in a round of iteration.

If a generalization gen ∈ CGSet, performing gen can probably change the anonymity of the data set, i.e., Ap (gen) – Ac (gen) be probably greater than 0. On the contrary, if gen ∈ NCGSet, Ap (gen) – Ac (gen)= 0., PG(gen) = 0. Assume all the generalization candidates are sorted ascending according to the ILPG values. It is possible to conduct all the candidates before the first critical generalization simultaneously, without affecting the anonymization result.

The first critical generalization can be performed in the same round definitely while others in RGSet possibly do this. Note that conducting critical generalizations potentially affects the ILPG values of candidates, thereby updating ILPG values is mandatory. To identify the available critical generalizations in one round from RGSet, a subroutine is presented in procedure 3.2. Let ACG denote the resultant available critical generalization set, i.e., the generalizations in ACGSet can be performed in one round of iteration together with the generalization before the first critical generalization. In the procedure, a priority queue is leveraged to keep the generalizations sorted ascendingly with respect to ILPG.

4.2.1 Procedure Indentifying available Generalizations

- Input Racing Generalization Set (RGSet), and Anonymity Quasi Identifier Set (AQI Set).
- Sort all the active generalization.
- Identify critical generalization.
- ACGset is got as the output.

Input parameter RGSet can be obtained readily after sorting all the active generalization candidates and identifying critical generalizations. Identifying CGSet relies on the other input parameter i.e., AQISet. Hence, identifying AQISet is a key to Procedure 4.2. In the following section, it is shown how to identify AQISet in the Map Reduce job ILPG Calculation. Once ACGSet is identified, it is possible to construct available generalization set AGSet, i.e., Set = ACGSet ∪ {gen}, where gen locates before the first critical generalization.

4.3 Information Loss per Privacy Gain Calculation Job

The Information Loss per Privacy Gain (ILPG) Calculation job is responsible for ILPG initialization in Step 2 and ILPG update in Step 3, 4 of procedure 4.1. The computation required in ILPG initialization is quite similar to that of ILPG update. The Map function of the ILPG Calculation is depicted in procedure 4.3, while the Reduce function is presented in procedure 4.4. In procedure 4.3 and procedure 4.4, the symbol '#' is used to identify whether a key is emitted to compute information gain or anonymity loss, and '$' is to differentiate the cases whether a key is for computing Ap (spec) or Ac (spec).

4.3.1 Procedure ILPG Calculation Map

1. Input data record (IDr , r ∈ D; anonymization level AL, NGSet.
2. For each attribute value vi in r, find its generalization in current AL.
3. If the generalization in current AL belongs to the NGSet , then it emits the key-value pair to the reduce function for information loss computation if this pair is a new generalization candidate.
4. Find the anonymity of the data set.
5. Gives the key-value pairs to obtain the anonymity after performing a generalization as output.

Procedure 4.3 shows the IGPL calculation of map and procedure 4.4 shows the IGPL calculation of reduce. The Reduce function described in Procedure 4.4 mainly aggregates the statistical information to calculate information loss and privacy gain. Step 1 and Step 2 calculate information loss. Due to that the key-value pairs are

sorted by Map Reduce built-in mechanism before being fed to Reducer workers, the Reduce function can compute information loss for generalizations in order, without requiring a large amount of memory to retaining statistical information. Therefore, the Reduce function is highly scalable for calculating information loss.

4.3.2 Procedure ILPG Calculation Reduce

- Input Intermediate key-pair ,key, list,count.
- For each key , initialize the sum of all counts to a variable sum.
- For each key update statistical count.
  - If all sensitive values for child c have arrived, compute I(Rc).
  - If all children c of parent p have arrived, compute I(Rp) and IL(gen) ; Emit (gen, IL(gen));
  - For each key update anonymity.
    - Update current anonymity
    - Update anonymity of generation.
- Information gain (gen, Ap(gen) and anonymity (gen, Ac (gen) , AQISet , (gen), Ap (gen) for generalizations are got as outputs.

The main step of computing anonymity of a data set is to find out the minimum QI-group size. Step 4 and Step 5 aims at calculating privacy gain as well as identifying AQISet. The Reducer workers find out the locally minimum QI-group size before and after performing a generalization in parallel. Then, it is possible to obtain the globally minimum QI-group size in the driver program through comparing the outputs of Reducer workers. The quasi-identifiers of the QI-groups with the minimum group size are recorded during the process and constitute AQISet. Note that AQISet plays an important role in identifying available generalizations in the next round of iteration. Above all, the ILPG Calculation Reduce function is highly scalable for both information loss and privacy gain computation. After obtaining information loss and privacy gain, ILPG values are calculated.

5. Evaluation

5.1 Overall Comparison

In this section, the effectiveness and efficiency of the proposed approach are empirically evaluated and compared with the existing state-of-the-art methods. Concretely, four groups of experiments are conducted for a comprehensive evaluation. In the first one, MRBUG is compared with traditional BUG, in terms of scalability and time-efficiency, to demonstrate the need for scalable methods for BUG when data sets are huge. Serial BUG method has been implemented according to procedural description in the above literature. In the second group of experiments, the scalability impacts of the k-anonymity parameter k on MRBUG is implemented, and quantitatively show the demand for the MRBUG approach that chooses a component according to the value of k . In the third group, the scalability and time-efficiency of the MRBUG approach with respect to the number data records is implemented, as the number of records dominates the time and space complexity of BUG. The effects of computation nodes on the scalability and time efficiency are reported in the fourth group.

The experiments are conducted on the amazon EC2 - Cloud platform. Adult data set and its enlarged versions have been used. All procedures are implemented in Java, and the Map Reduce implementation is based on Hadoop 1.0.0 Map Reduce APIs. The execution time of methods is measured for the scalability and time-efficiency. The data distortion is captured by ILoss. The value of ILoss is normalized to facilitate comparisons. Each round of experiment is repeated ten times. The mean and standard errors of measured results are reported for a comprehensive evaluation.

5.2 Comparison with Serial BUG

To show the need for scalable procedures, MRBUG is compared with their traditional counterparts, i.e., serial BUG. The number of records ranges from 500,000 to 5,000,000. Thus, the data sets in these experiments are big enough to evaluate the effectiveness of this approach in terms of the number of data records. The k - anonymity parameter is set as fifty. The value of k is selected randomly and does not affect the analysis in this group of experiments, as what is to be observed is the scalability changes of serial and Map Reduce based procedures with respect to the number of records. Interesting readers can try other values. The conclusions will be the same. As the Map Reduce methods incur the same amount of data distortion as its serial counterparts, only the results of execution time is presented. To make a fair comparison, the serial procedures are executed on a virtual machine of m1.large type that has four virtual CPUs and

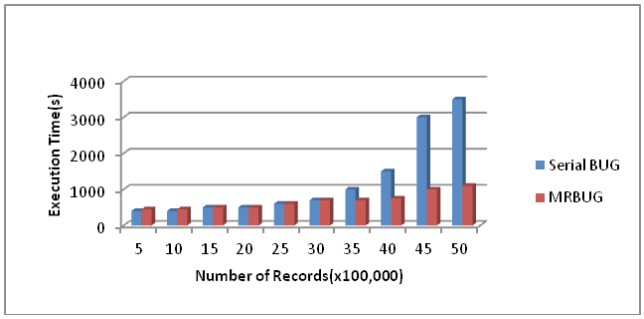


Figure 1. Change of execution time w.r.t. number of records. (Serial BUG Vs. MRBUG).

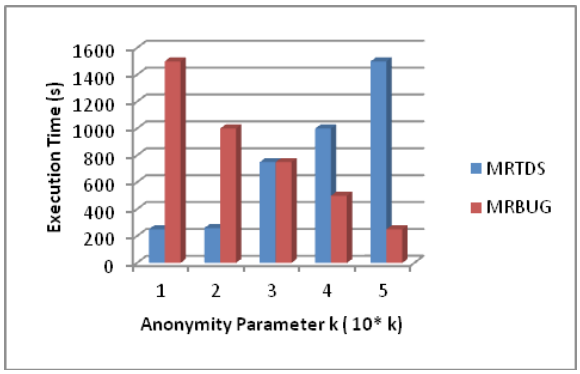
8 GB memory, while Map Reduce based procedures are executed on a cluster that consists of 10 virtual machines of m1.medium type having two virtual CPUs and four GB memory. Figure 1 shows the change of execution time with respect to number of records in serial BUG and MRBUG.

The execution time of serial BUG grows gradually at former stages, but goes up severely at later stages. It runs out of memory when the number of records researches to 5,000,000, since the indexing data structure consumes too much memory. The execution time of MRBUG increases relatively slowly and smoothly. Its scalability will be further evaluated subsequently. The above experimental results demonstrate the insufficiency of the state-of-the art subtree anonymization approaches when handling large data sets. Hence, it is necessary to propose scalable BUG procedures for large data.

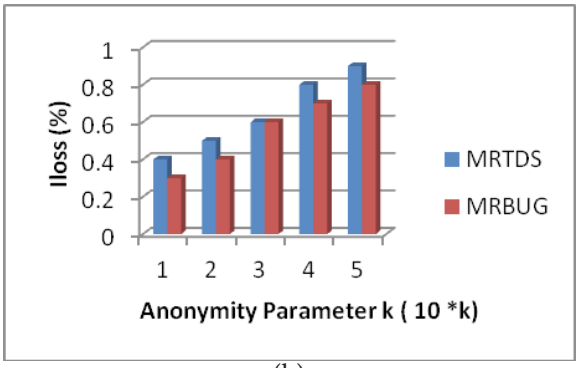
5.3 Scalability of MRTDS Vs. MRBUG

In this group of experiments, the impacts of the anonymity parameter k on the scalability of MRTDS and MRBUG is examined. The number of data records is set as 1,000,000. In fact, k can be valued from 1 to 1,000,000, where k= 1 and k= 1,000,000 are two extreme cases. In terms of the generalization process, the anonymity of a data set varies in an exponential manner. To comply with this fact, the values of k in the form of 10x are selected for comprehensive evaluation, where k is a non-negative integer. Specifically, k ranges from one to five. The number of Reducers is set as ten. The experiment results are reported in Figure 1. To present the results in a manageable and intuitive way, the horizontal axis is logarithmically scaled with base ten.

Figure 2(a) shows the change of execution time with respect to k for MRTDS and MRBUG. The execution



(a)



(b)

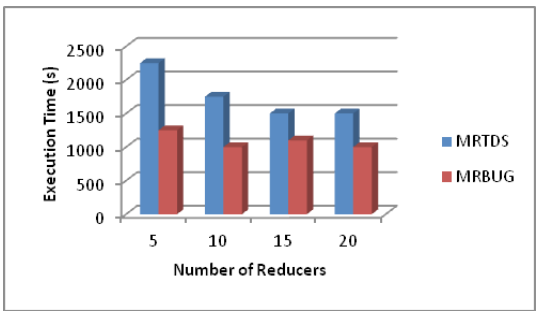
Figure 2. Changes of execution time and ILoss % w.r.t. anonymity parameter k (MRTDS Vs. MRBUG).

time of MRTDS decreases stably linearly when the orders of magnitude of k grows. On the contrary, the execution time increases approximately linearly when k is getting large. The two curves intersect at the middle point of k's orders of magnitude. MRBUG takes less execution time before this point, and the smaller k is, the less it takes than MRTDS. After this point, MRTDS consumes less time, and the larger k is, the less time it consumes. Figure 2(b) demonstrates the change of ILoss% with respect to k for MRTDS and MRBUG. The ILoss% of both methods increase with the growing of k, which reflects the fact that larger k implies more data distortion. Because, the resultant anonymization levels may be different for small k. The data distortions caused by MRTDS and MRBUG are different. But the differences are minor according to the results in Figure 2(b). This trend ensures that MRBUG can be utilized freely without considering the data distortion aspect. According to the tendencies of execution time and ILoss, it is necessary and reasonable to choose MRBUG to perform parallel generalized data anonymization for large data according to the value of k. This empirically validates

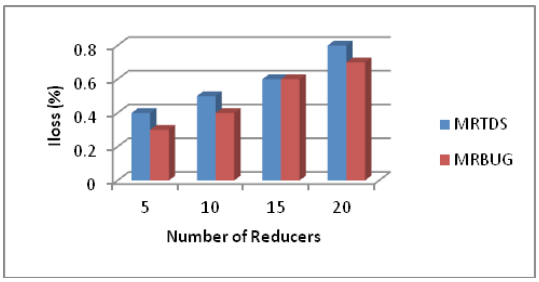
the motivation of the MRBUG approach. The scalability of the hybrid approach is evaluated next.

5.4 Scalability of MRBUG over Computation Nodes

Another aspect of scalability evaluation is to explore whether the approach is scalable over computation nodes. The number of Reducers ranges from five to twenty. Each computation node is of the m1.medium type. The number of data records in this group of experiments is set as 1,000,000. Similar to the last group of experiments, the k-anonymity parameter is set as 100 and 1000 for MRBUG. Figure 3 demonstrates the execution time and ILoss of the worst case of the MRBUG approach. It can be seen from Figure 3(a) that the execution time of MRTDS drops off in a nearly linear fashion when the number of Reducers is getting larger. This illustrates that the MRBUG approach is linearly scalable with respect to the number of Reducers. As a result, the proposed approach can handle large data sets with ease by just employing more computation nodes. Figure 3(b) demos that the ILoss keeps constant with the increase of Reducers. This is reasonable as the ILoss is affected by anonymity parameter k and the data set, but the number of Reducers. The above four sets of experiments reveals that the parallel BUG approach can significantly improve the privacy preservation with minimum information loss over large data set compared with the existing serial BUG approaches.



(a)



(b)

Figure 3. Change of execution time and ILoss w.r.t number of reducers.

6. Conclusions and Future Work

In this paper, the scalability problem of large-scale data anonymization by Bottom-Up Generalization (BUG), and proposed a highly scalable parallel BUG approach using Map reduce on cloud has been investigated. Datasets are partitioned and anonymized in parallel in the first phase, producing intermediate results. Then, the intermediate results are merged and anonymized to produce consistent k-anonymous data sets in the second phase.

Map reduce technique has been creatively applied on cloud to data anonymization and formulated a group of innovative Map reduce jobs to achieve Generalization computations in a highly scalable way. Experimental results on real-world datasets have revealed that with this approach, BUG is scalable and efficient than any other approach. In cloud environment, the privacy preservation for data analysis, sharing and mining is a challenging research issue because gradually larger volumes of datasets are used, thereby demanding severe research. A thorough investigation is done with the bottom-up generalization algorithms for data anonymization. Based on the contributions herein, it is intended to explore the subsequent phase on scalable privacy preservation aware analysis and scheduling on large-scale datasets. Optimized, heuristic, and balanced scheduling approaches are expected to be developed towards overall scalable privacy preservation. It is believed that the structure of bottom-up generalization is amenable to several extensions that make it more practical. Incorporating different metrics and handling data suppressions in partial generalization is not necessarily require to have all child values generalized altogether. It is also possible to generalize numeric attributes without a pre-determined hierarchy and shall be taken up as a future work.

7. References

1. Chaudhuri S. What next? A half-dozen data management research goals for big data and the cloud.Proceedings of the 31st Symposium on Principles of Database Systems (PODS'12); 2012. p. 1–4.
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, et al. A view of cloud computing. Communication of ACM. 2010; 53(4):50–8.

3. Wang L, Zhan J, Shi W, Liang Y. In cloud, may scientific communities benefit from the economies of scale? IEEE Trans Parallel Distrib Syst. 2012; 23(2):296–303.
4. Takabi H, Joshi JBD, Ahn G. Security and privacy challenges in cloud computing environments. IEEE Security and Privacy. 2010; 8(6):24 – 31.
5. Zissis D, Lekkas D. Addressing cloud computing security issues. Future General Computer Systems. 2011; 28(3): 583 –92.
6. Zhang X, Liu C, Nepal S, Pandey S, Chen J. A privacy leakage upper-bound constraint based approach for cost-effective privacy preserving of intermediate datasets in cloud. IEEE Transactions Parallel Distributed Systems. 2012 Aug 08; 24(6):1192–202.
7. Hsiao-Ying L, Tzeng WG. A Secure erasure code-based cloud storage system with secure data forwarding. IEEE Transactions Parallel Distributed Systems. 2012; 23(6):995–1003.
8. Cao N, Wang C, Li M, Ren K, Lou W. Privacy-preserving multi-keyword ranked search over encrypted cloud data. Proceedings 31st Annual IEEE International Conference on Computer Communications (INFOCOM'11); 2011 Apr 10-15. p. 829–37.
9. Mohan P, Thakurta A, Shi E, Song D, Culler D. Gupt: Privacy preserving data analysis made easy. Proceedings of ACMSIGMOD International Conference on Management of Data (SIGMOD'12); 2012. p. 349–60. Available from: http://www.microsoft.com/health/ww/products/Pages/healthvault.aspx
10. Fung BCM, Wang K, Chen R, Yu PS. Privacy-preserving data publishing: A survey of recent developments. ACM Comput Survey. 2010; 42(4):1 –53.
11. Fung BCM, Wang K, Yu PS. Anonymizing classification data for privacy preservation. IEEE Transaction Knowledge Data Engineer. 2007 Mar 26; 19(5):711 –25.
12. Xiao X, Tao Y. Anatomy: Simple and effective privacy preservation. Proceedings of 32 International Conference on Very Large Data Bases (VLDB'06); 2006. p. 139–50.
13. LeFevre K, DeWitt DJ, Ramakrishnan R. Incognito: Efficient full-domain K-anonymity. Proceedings of 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD'05); 2005. p. 49–60.
14. LeFevre K, DeWitt DJ, Ramakrishnan R. Mondrian multi-dimensional K-anonymity. Proceedings 22nd International Conference on Data Engineering (ICDE'06); 2006.
15. Borkar V, Carey MJ, Li C. Inside big data management: Ogres, Onions, or Parfaits? Proceedings 15th International Conference Extending Database Technology (EDBT'12); 2012. p. 3-14.
16. LeFevre K, DeWitt DJ, Ramakrishna R. Workload-aware anonymization techniques for large-scale datasets. ACM Transcations Database System. 2008 Aug 17; 33(3):1–47.
17. Iwuchukwu T, Naughton JF. K-anonymization as spatial indexing: Toward scalable and incremental anonymization. Proceedings 33rd International Conference on Very Large Data Bases (VLDB'07); 2007. p. 746–57.
18. Dean J, Ghemawat S. Map Reduce: Simplified data processing on large clusters. Communications ACM. 2008; 51(1):107–13.
19. Mohammed N, Fung B, Hung PCK, Lee CK. Centralized and distributed anonymization for high-dimensional healthcare data. ACM Transcations Knowledge Discover Data. 2010 Oct; 4(4): 1–33.
20. Fung B,Wang K,Wang L,Hung PCK. Privacy-Preserving Data Publishing for Cluster Analysis: Data Knowledge Engineering. 2009; 68(6): 552–75. doi:10.1016/j.datak.2008.12.001.
21. Mohammed N, Fung BC, Debbabi M. Anonymity meets game theory: secure data integration with malicious participants. VLDB J. 2011 Aug; 20(4):567–88.
22. Agrawal R, Srikant R. Privacy preserving data mining. Proceedings of Special Interest Group on Management of Data (SIGMOD); 2000 Jun. p. 439–50.
23. Hundepool, Willenborg L. μ- and -argus: Software for statistical disclosure control. Proceedings 3rd International Seminar on Statistical Confidentiality; Bled; 1996.
24. Sweeney L. Achieving K-anonymity privacy protection using generalization and suppression. Proceedings International Journal on Uncertainty, Fuzziness and Knowledge-based Systems. 2002; 10(5):571–88.
25. Sweeney L. K-anonymity: A model for projecting privacy. Proceedings International Journal on Uncertainty, Fuzziness and Knowledge-based Systems. 2002 Oct; 10(5):557–70.
26. Jiang W, Clifton C. A secure distributed framework for achieving K-anonymity. VLDB J. 2006; 15(4):316–33.
27. Jurczyk P, Xiong L. Distributed anonymization: Achieving privacy for both data subjects and data providers. Data and Applications Security XXIII Database Section (DBSec'09); 2009. p. 191–207.
28. Roy STVS, Kilzer A, Shmatikov V, Witchel E. Airavat: Security and privacy for Map Reduce. Proceedings 7USENIX Conference Networked Systems Design and Implementation (NSDI'10); p. 297–312.
29. Zhang K, Zhou X, Chen Y, Wang X, Ruan Y. Sedic:-Privacy-Aware data intensive computing on hybrid clouds. Proceedings of 18th ACM Conference on Computer and Communications Security (CCS'11); 2011. p. 515–26.
30. Xiao X, Tao Y. Personalized privacy preservation. Proceedings 2006 ACM SIGMOD International Conference Management of Data (SIGMOD'06); 2006. p. 229–40.