A Bigdata Sets Approach for Data Anonymization Using Map-Reduce on Cloud

T.Bharathi, Sri K.Sreenivasulu
Department of Computer science and Engineering, Madina Engineering College, JNTUA University, India

Abstract

Most cloud services require users to share personal data like electronic health records for analysis of data (or) mining, bringing privacy concerns. Generally cloud computing refers the practice of using a network of remote servers hosted on the internet store manage and process the data rather than local server (or) PC. In many cloud applications at present the scale of data increases in accordance with BIG DATA there by making it a complicated to commonly used software tools to handle and process a large-scale data with a elapsed time. It is challenging for previous anonymization approaches to achieve privacy preservation on large-scale data sets due to insufficiency.

The proposed a scalable 2-phaseTDS uses a map reduce architecture on cloud to anonym zed large-scale datasets finally a group of innovative map reduce jobs. So efficiency 2-phaseTDS can be significantly improved over existing approaches. The technique of map reduce method will be applied to distributed computing & parallel computing.

Keywords: Data anonymization, 2P-TDS, Map-Reduce Architecture, privacy preservation.

I. INTRODUCTION

The process of data publication is becoming larger and complex day by day. Cloud computing is the most popular model for supporting large and complex data, most organizations are moving towards to reduce their cost and elasticity features. However cloud computing has potential risk and vulnerabilities. One of major problem in moving to cloud computing is its security and privacy concerns. Cloud computing provides powerful and economical infrastructural resources for cloud users to handle ever increasing data sets in big data applications. However, processing or sharing privacy-sensitive data sets on cloud probably leads to privacy concerns because of multi-tenancy system. Data encryption and anonymization is two widely-adopted ways to combat privacy breach. The encryption is not suitable for data that are processed and shared frequently and the anonymizing big data and manages anonymized data sets are still challenges for traditional anonymization approaches. Thus, various proposals have been designed in a cloud computing for privacy preserving in data publishing. In this paper, we survey the current existing techniques, and analyze the advantage and disadvantage of these approaches. In this paper, we propose a highly scalable two-phase TDS approach for data anonymization based on Map-Reduce on cloud. To make full use of the parallel capability of Map-Reduce 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 datasets, 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 datasets. We leverage Map-Reduce to accomplish the concrete computation in both phases. A group of Map-Reduce jobs is deliberately designed and coordinated to perform specializations on data sets collaboratively. We evaluate our approach by conducting experiments on real-world datasets. Experimental results demonstrate that with this approach, the scalability and efficiency of TDS can be improved significantly over existing approaches.

II. PROPOSED SYSTEM

In proposed system we design the 2P-TDS with MAP-REDUCE, Data Anonymization, and privacy preservation.

A. DATA ANNOYMIZATION

Technology that converts clear text data into a nonhuman readable and irreversible form, including but not limited to pre-image resistant hashes and encryption techniques in which the decryption key has been discarded. Data is considered anonymized even when conjoined with pointer or pedigree values that direct the user to the originating system, record, and value (e.g., supporting selective revelation) and when anonymized records can be associated, matched, and/or conjoined with other anonymized records.
B. K-ANONYMITY

The goal is to make each record indistinguishable from a defined number (k) other records, if attempts are made to identify the record. k-anonymity guarantees that each sensitive attribute is hidden in the scale of k groups. This means that the probability of recognizing the individual does not exceed 1/k. The level of privacy depends on the size of k. The statistical characteristics of the data are retained as much as possible; however, k-anonymity is not only applicable to sensitive data. An attacker could mount a consistency attack or background-knowledge attack to confirm a link between sensitive data and personal data. This would constitute a breach of privacy. The extensive study resolved some shortcomings of k-anonymity model as listed below.

1) It can’t resist a kind of attack, which is assuming that the attacker has background knowledge to rule out some possible values in a sensitive attribute for the targeted victim. That is, k-anonymity does not guarantee privacy against attackers using background knowledge. It is also susceptible to homogeneity attack. An attacker can discover the values of sensitive attributes when there is little diversity in those sensitive attributes. Thus some stronger definitions of privacy are generated, such as l-Diversity.

2) It protects identification information. However, it does not protect sensitive relationships in a data set.

3) Although the existing k-anonymity property protects against identity disclosure, it fails to protect against attribute disclosure.

4) It is suitable only for categorical sensitive attributes. However, if we apply them directly to numerical sensitive attributes (e.g., salary) may result in undesirable information leakage.

5) It does not take into account personal anonymity requirements and a k-anonymity table may lose considerable information from the micromanagement for the allocation of public funds, medical research, and trend analysis.

Direct Anonymization Algorithm DA (D, J, k, m)

1. Scan D and create count-tree
2. Initialize C-out
3. For each node v in preorder count-tree transversal do
   4. If the item of v has been generalized in C-out then
     5. Backtrack
   6. If v is a leaf node and v.count < k then
     7. J = item set corresponding to v
     8. Find generalization of items in J that make J k-anonymous
   9. Merge generalization rules with C-out
   10. Backtrack to longest prefix of path J, wherein no item has been generalized in C-out
   11. Return C-out
     12. for i := 1 to Count do
     13. Initialize count = 0
     14. Scan each transaction in C-out
     15. Separate each item in a transaction and store it in p
     16. Increment count
     17. for j := 1 to count do
     18. For all g belongs C-out do
     19. Compare each item of p with that of C-out
     20. If all items of i equal to c-out
     21. Increment the r
     22. If k a equal to r then backtrack to i
     23. Else if r greater than ka then get the index position of the similar transactions
     24. Make them NULL until ka equal to r
     25. Else update the transactions in database

B. 2P-TDS

A top-down approach is essentially the breaking down of a system to gain insight into its compositional sub-systems. In a top-down approach an overview of the system is formulated, specifying but not detailing any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements. A top-down model is often specified with the assistance of "black boxes", these make it easier to manipulate. However, black boxes may fail to elucidate elementary mechanisms or be detailed enough to realistically validate the model. Top down approach starts with the big picture. It breaks down from there into smaller segments. Specialization is an important way to generate propositional knowledge, by applying general knowledge, such as the theory of gravity, to specific instances, such as "when I release this apple, it will fall to the floor". Specialization is the opposite of generalization. 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 Map-Reduce on cloud. Basically, Map-Reduce on cloud have two levels of parallelization, i.e., job level and task level. Job level parallelization means that multiple Map-Reduce jobs can be executed simultaneously to make full use of cloud infrastructure resources. Combined with cloud, Map-Reduce become more powerful and elastic as cloud can offer infrastructure resources on demand, for example.
C. MAP-REDUCE ARCHITECTURE

Map-Reduce is the programming model for processing large data sets with a parallel, distributed algorithm on a cluster. A Map-Reduce program is composed of a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "Map-Reduce System" (also called "infrastructure", "framework") orchestrates by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, providing for redundancy and fault tolerance, and overall management of the whole process.

The model is inspired by the map and reduce functions commonly used in functional programming, although their purpose in the Map-Reduce framework is not the same as their original forms. Furthermore, the key contribution of the Map-Reduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once.

Map-Reduce libraries have been written in many programming languages, with different levels of optimization. A popular open-source implementation is Apache Hadoop. The name Map-Reduce originally referred to the proprietary Google technology and has since been generalized.

"Map" step: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node.

"Reduce" step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve.

Map-Reduce allows for distributed processing of the map and reduction operations. Provided each mapping operation is independent of the others, all maps can be performed in parallel – though in practice it is limited by the number of independent data sources and/or the number of CPUs near each source. Similarly, a set of 'reducers' can perform the reduction phase - provided all outputs of the map operation that share the same key are presented to the same reducer at the same time, or if the reduction function is associative.

5-step parallel and distributed computation:

- Prepare the Map() input – the "Map-Reduce system" designates Map processors, assigns the K1 input key value each processor would work on, and provides that processor with all the input data associated with that key value.
- Run the user-provided Map() code – Map() is run exactly once for each K1 key value, generating output organized by key values K2.
- "Shuffle" the Map output to the Reduce processors – the Map-Reduce system designates Reduce processors, assigns the K2 key value each processor would work on, and provides that processor with all the Map-generated data associated with that key value.
- Run the user-provided Reduce () code – Reduce () is run exactly once for each K2 key value produced by the Map step.
- Produce the final output – the Map-Reduce system collects all the Reduce output, and sorts it by K2 to produce the final outcome.

ALGORITHM:

Map (key, value):

// key: document name; value: text of document for each word w in value:
Emit (w, 1)

Reduce (key, values):

// key: a word; values: an iterator over counts
Result = 0
for each count v in values:
Result += v
Emit (key, result)
III. RESULT ANALYSIS

The experiments are unit conducted during cloud surroundings named U-Cloud. U-Cloud could be a cloud computing surroundings at the University of Technology state capital (UTS). The system summary of U-Cloud has been delineated in the computing facilities of this method area unit set among many labs at UTS. On high of hardware and UNIX system operating system (Ubuntu), we have a tendency to install KVM virtualization software that virtualizes the infrastructure and provides unified computing and storage resources virtualized data centers, we have a tendency to install Open-Stack open supply cloud environment for international management, resource programing and interaction with users. Further, Hadoop clusters are engineered supported the Open-Stack cloud platform to facilitate large-scale processing. Both TPTDS and Cent-TDS area unit enforced in Java.

IV. CONCLUSION

In this, we have investigated the scalability problem of large-scale data anonymization by TDS, and proposed a highly scalable two-phase TDS approach using Map-Reduce 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.

I have creatively applied Map-Reduce on cloud to data anonymization and deliberately designed a group of innovative Map-Reduce 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.

V. FUTURE ENHANCEMENT

A possible way of data anonymization in which the situation may be improved and next generation solutions may be developed. As future work a combination of top-down and bottom up approach generalization is contributed for data anonymization in which data Generalization hierarchy is utilized for anonymization.

REFERENCES

