RNI Regd No: APENG/2011/47294 ISSN No: 2320-3706(PRINT)
International Journal & Magazine of Engineering, Technology, Management and Research
YUVAENGINEERS
Transforming Young Engineers for Better Tomorrow

The Scalability Problem of Large-Scale Data Anonymization by TDS

M. Sunitha
Assistant Professor,
Department of Computer Science,
Malla Reddy Engineering College for Women,
Maisammaguda, Hyderabad.

Glacy Elizabeth Jacob
Assistant Professor,
Department of Computer Science,
Malla Reddy Engineering College for Women,
Maisammaguda, Hyderabad.

Abstract:

A large number of cloud services requires users to share private data like electronic health records for data analysis or mining, bringing privacy concerns. Anonymzing 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 tremendously in accordance with the Big Data trend, thereby making it a challenge for commonly used software tools to capture, manage and process such large-scale data within a tolerable elapsed time. As a result is challenge for existing Anonymization approaches to achieve privacy preservation on privacy-sensitive large-scale data sets due to their insufficiency of scalability. An introduce the scalable two-phase top-down specialization approach to anonymized large-scale data sets using the Map Reduce framework on cloud.

In both phases of approach is deliberately design a group of innovative Map Reduce jobs to concretely accomplish the specialization computation in a highly scalable way. Experimental evaluation results demonstrate that with this approach. The scalability and efficiency of top-down specialization can be improved significantly over existing approaches. An introduce the scheduling mechanism called Optimized Balanced Scheduling to apply the Anonymization. Here the OBS means individual dataset have the separate sensitive field. Every data set sensitive field and give priority for this sensitive field. Then apply Anonymization on this sensitive field only depending upon the scheduling.

Key Words:

Data Anonymization, Top-Down specialization, Map-Reduce, Cloud, Privacy Preservation, OBS, Data Partition, Data Merging.

INTRODUCTION:

Cloud computing act as major impact on present IT industry and research communities. Cloud computing provides a great computation power and storage capacity through the use of cast number of computers together, that enable users to deploy their application in a very profitable way. Even though, the cloud can reduce the user contribution on IT infrastructure, several customers are still afraid to take use of cloud because of privacy and security issues. Data privacy can be easily hefted by malicious cloud users and providers because of some failures in the commonly used privacy protection techniques on cloud. This can cause substantial economic loss or severe social influence deterioration to data owners, hence data privacy is one of the most concerned issues that need to be addressed imperatively before the data sets are analyzed and shared on cloud. Data Anonymization technique is widely take up for preserving privacy of data that are published and shared among the users on cloud.

Data Anonymization is the process of hiding identity or more sensitive data that are stored on the cloud. Then the privacy of that data sets can be achieved effectively. For that purpose, a different kind of Anonymization algorithms with different Anonymization operations have been proposed. Now a days the scale of data sets that are stored and shared between the customers in some cloud application increases rapidly therefore , it is a challenge for existing Anonymization algorithms to anonymized such large scale data sets. Map reduce framework have been adopted for handling and processing such large scale-data and that provides greater computation capability for applications. Such frameworks are appropriate to address the scalability problem of Anonymzing large scale data sets. Map reduce framework is widely adopted for parallel data processing ,to address the scalability problem of the top-down specialization(TDS) approach for large scale data Anonymization.

TDS approach is widely used for data Anonymization that provides a good arbitrate between data utility and data consistency. Most of the TDS algorithm are centralized, that are insufficient to handle large-scale data sets. Therefore some distributed algorithms have been proposed. In this paper, we introduce a highly scalable two phase TDS approach for data Anonymization by using the map reduce framework on cloud.The Anonymization process in the map reduce framework can be divided into two phases. In the first phase the original data sets splited into smaller group of data sets, and these data sets are anonymized parallel and thus producing a intermediate results. In the second phase the intermediate results are merged together and further anonymized the joined data sets to achieve consistent k-anonymous data sets. A group of map reduce jobs are designed and coordinated to perform specializations on data sets.

LITERATURE SURVEY:
DATA ANONYMIZATION:

Technology that converts clear text data into a nonhuman readable and irreversible form, including but not limited to pre-image resistant hashes (e.g., one-way 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. Data Anonymization enables the transfer of information across a boundary, such as between two departments within an agency or between two agencies, while reducing the risk of unintended disclosure, and in certain environments in a manner that enables evaluation and analytics post-Anonymization.

TOP DOWN APPROACH:

A top-down approach (also known as stepwise design and in some cases used as a synonym of decomposition) 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:

Specializations, (or 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". Specializations is the opposite of generalization.

MAPREDUCE:

Map Reduce is a 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.

Map Reduce is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Computational processing can occur on data stored either in a file system (unstructured) or in a database (structured). Map Reduce can take advantage of locality of data, processing data on or near the storage assets to decrease transmission of data.

"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. While this process can often appear inefficient compared to algorithms that are more sequential, Map Reduce can be applied to significantly larger datasets than "commodity" servers can handle – a large server farm can use Map Reduce to sort a peat byte of data in only a few hours.[citation needed]The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled – assuming the input data is still available.

Another way to look at Map Reduce is as a 5-step parallel and distributed computation:

1.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.

2.Run the user-provided Map() code – Map() is run exactly once for each K1 key value, generating output organized by key values K2.

3."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.

4.Run the user-provided Reduce() code – Reduce() is run exactly once for each K2 key value produced by the Map step.

5.Produce the final output – the Map Reduce system collects all the Reduce output, and sorts it by K2 to produce the final outcome.

RELATED WORK:

Recently, data privacy preservation has been extensively investigated . We briefly review related work below. Lefebvre et al. addressed the scalability problem of Anonymization algorithms via introducing scalable decision trees and sampling techniques.

Iwuchukwu and Naught on 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 , there by failing to work in the TDS approach. Fung teal. , proposed the TDS approach that produces anonymous data sets without the data exploration problem . 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 and Mohammed et al. proposed distributed algorithms to anonymized vertically partitioned data from different data sources without disclosing privacy information from one party to another.

Jurczyk and Xing and Mohammed et al. proposed distributed algorithms to anonymized horizontally partitioned data sets retained by multiple holders. However, the above distributed algorithms mainly aim at securely integrating and Anonymzing 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. 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. leveraged MapReduce to automatically partition a computing job in terms of data security levels, protecting data privacy in hybrid cloud. Our research exploits MapReduce itself to anonymized large-scale data sets before data are further processed by other MapReduce jobs, arriving at privacy preservation.

PROBLEM STATEMENT:

A widely adopted parallel data processing framework, to address the scalability problem of the top-down specialization (TDS) approach for large-scale data Anonymization. The TDS approach, offering a good tradeoff between data utility and data consistency, is widely applied for data Anonymization. Most TDS algorithms are centralized, resulting in their inadequacy in handling large-scale data sets. Although some distributed algorithms have been proposed, they mainly focus on secure Anonymization of data sets from multiple parties, rather than the scalability aspect.

DRAWBACKS:

•The Map Reduce computation paradigm still a challenge to design proper MapReduce jobs for TDS.
•The overall performance of the privacy provided is low.

•It is only suitable for the small amount of data sets.
•The Anonymization of the each level is low.

PROBLEM DEFINITION:

•In this paper, we propose a scalable two-phase top-down specialization (TDS) approach to anonymized 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.

ADVANTAGES:

•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 data 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.

BLOCK DIAGRAM:

IMPLEMENTATION:
DATA PARTITION:

•In this module the data partition is performed on the cloud.
•Here we collect the large no of data sets.
•We are split the large into small data sets.
•Then we provides the random no for each data sets.

ANONYMIZATION:

•After getting the individual data sets we apply the Anonymization.
•The Anonymization means hide or remove the sensitive field in data sets.
•Then we get the intermediate result for the small data sets
•The intermediate results are used for the specialization process.
•All intermediate Anonymization levels are merged into one in the second phase. The merging of Anonymization levels is completed by merging cuts. To ensure that the merged intermediate Anonymization level ALI never violates privacy requirements, the more general one is selected as the merged one.

MERGING:

•The intermediate result of the several small data sets are merged here.
•The MRTDS driver is used to organizes the small intermediate result
•For merging, the merged data sets are collected on cloud.
•The merging result is again applied in Anonymization called specialization.

SPECIALIZATION:

•After getting the intermediate result those results are merged into one.
•Then we again applies the Anonymization on the merged data it called specialization.
•Here we are using the two kinds of jobs such as IGPL UPDATE AND IGPL INITIALIZATION.
•The jobs are organized by web using the driver.

OBS:

•The OBS called optimized balancing scheduling.
•Here we focus on the two kinds of the scheduling called time and size.
•Here data sets are split in to the specified size and applied Anonymization on specified time.

•The OBS approach is to provide the high ability on handles the large data sets.

METHODOLOGY:

A MapReduce program is consists of a Map() procedure that performs filtering and sorting (such as sorting students by email into queues, one queue for each email) and a Reduce() procedure that performs a summary operation (such as counting the number of students in every queue, resulting name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, executing the various tasks in parallel, keeping all communications and data transfers between the various parts of the system, and giving for redundancy and fault tolerance. The model is inspired by the map and reduces functions commonly used in programming, even though their purpose in the MapReduce framework is not the same as in their original forms.

The main contributions of the MapReduce framework are not the actual map and reduce functions, but the extensibility and fault-tolerance gained for a variety of applications by optimizing the execution engine once. A single-threaded implementation of MapReduce will usually not be faster than a traditional implementation. When the optimized distributed shuffle operation (which reduces network communication cost) and fault tolerance features of the MapReduce framework come into play, is the use of this model beneficial. MapReduce libraries have been written in multiple programming languages, with separate levels of optimization. A famous open-source implementation is Apache Hadoop. The name MapReduce originally referred to the proprietary Google technology but has since been generalized. The Hadoop distributed file system (HDFS) is a scalable, distributed and portable file-system written in Java for the Hadoop framework.

A Hadoop cluster has typically a single name node plus a cluster of data nodes, redundancy options are available for the name node due to its importance. Each data node serves blocks of data over the network using a block protocol specific to HDFS. The file system makes use of TCP/IP sockets for communication. Clients use RPC(remote procedure call) to communicate between each other. HDFS stores large files (typically in the range of gigabytes to terabytes) across no of machines. It achieves reliability by replicating the data across hosts, and hence theoretically does not need RAID storage on hosts (but to improve I/O performance some RAID configurations are still useful). With default replication value, 3, data is stored on three nodes: two on the same rack, and one on a separate rack. Data nodes can communicate with each other to adjust data, to move copies around, and to keep the replication of data.

HDFS is not POSIX-compliant, since the requirements for a POSIX file-system differ from the target goals for a Hadoop application. The advantage of not having a fully POSIX-compliant file-system is increased performance for data throughput and support for non-POSIX operations such as Append. First, many existing clustering algorithms (e.g., k- means) requires the calculation of the "cancroids". But there is no notion of "cancroids" in our setting where each attribute for mesa data point in the clustering space. Second, k-medic method is so reliable to the existence of outliers (i.e., data points that are very far away from the rest of data points). Third, the order in which the data points are examined does not affect the clusters computed from the kmedoidmethod. Existing Anonymization algorithms can be used for column generalization e.g. Mondrian. The algorithms can be verified on the sub table containing only attributes in one column to ensure the anonymity requirement. Existing data analysis (e.g., query answering) methods can be easily used on the sliced data. Current privacy measures for membership disclosure protection include differential privacy and presence.

CONCLUSIONS:

The conclusion of the proposed work is it using the two phase top down approach to provide ability to handles the high amount of the large data sets. And here it provide the privacy by effective Anonymization approaches. In the future work is reduce the handling effect of large amount of the data sets. Its implements the optimized balancing scheduling. Where it's based on the time and size of the data sets. In this paper it have investigated the scalability problem of large-scale data Anonymization by Top-Down Specialization and proposed a highly scalable two-phase TDS approach using Map Reduce on cloud. Datasets 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. It 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 datasets have demonstrated that with our approach, the scalability and efficiency of TDS are improved significantly over existing approaches.

FUTURE WORK:

It does not have the ability for handle the large scale datasets in cloud. Its overcome by we invent the two phase top-down specialization approach. 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. Here the draw backs of proposed system is there is no priority for applying the ANONYMIZATION on datasets. So that its take more time to ANONYMIZE the datasets. So we introduce the scheduling mechanism called OPTIMIZED BALANCED SCHEDULING(OBS) to apply the ANONYMIZATION. Here the OBS means individual dataset have the separate sensitive field. 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.

REFERENCES:

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

[2] M. Armrest, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Kaminski, G. Lee, D. Patterson, A. Rabin, I. Stoical, and M. Zaharias, "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. Taka, J.B.D. Joshi, and G. An, "Security and Privacy Challenges in Cloud Computing Environments," IEEE Security and Privacy, vol. 8, no. 6, pp. 24-31, Nov. 2010.

[5] D. Sissies and D. Lukas, "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. Pander, 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. Ten, "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. Ran, 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, "Gut: Privacy Preserving Data Analysis Made Easy," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '12), pp. 349- 360, 2012.

[10] Microsoft Health Vault, 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 Devil- opens," ACM Computing Surveys, vol. 42, no. 4, pp. 1-53, 2010.

[12] B.C.M. Fung, K. Wang, and P.S. Yu, "Anonymzing 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. Lefebvre, D.J. DeWitt, and R. Ramakrishna, "Incognito: Efficient Full-Domain K-Anonymity," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '05), pp. 49-60, 2005.

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

[16] V. Broker, 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. Lefebvre, D.J. DeWitt, and R. Ramakrishna, "Workload-Aware Anonymization Techniques for Large-Scale Data Sets," ACM Trans. Database Systems, vol. 33, no. 3, pp. 1-47, 2008.