ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The Scalability Problem of Large-Scale Data Sets Using the Map Reduce Framework on Cloud

Apathi Haripriya,
M.Tech,
Department of CSE,
Global Institute of Engineering and Technology,
Chilkur(V), RR District, Telangana

Mrs. M.Jhansi Lakshmi,
Associate professor,
HoD of CSE,
Global Institute of Engineering and Technology,
Chilkur(V), RRDistrict, Telangana

ABSTRACT

A large number of cloud services requires users to share private data like electronic health records for the data analysis or mining, bringing privacy concerns. Unidentified datasets 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 unidentification 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 unidentified 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 introducing the scheduling mechanism called Optimized Balanced Scheduling to apply the Unidentification. 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 Unidentification on this sensitive field only depending upon the scheduling.

Key Words: ***Data Unidentification, Top-Down specialization, Map-Reduce, Cloud, Privacy Preservation, OBS, Data Partition, Data Merging.***

INTRODUCTION

Distributed computing is a field of computer science that studies distributed systems. A distributed system is a software system in which components located on networked computers communicate and coordinate their actions by passing messages.[1] The components interact with each other in order to achieve a common goal. There are many alternatives for the message passing mechanism, including RPC-like connectors and message queues. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components.[1] An important goal and challenge of distributed systems is location transparency. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications. A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs. Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate with each other by message passing. The word distributed in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

processes that run on the same physical computer and interact with each other by message passing. While there is no single definition of a distributed system,[6] the following defining properties are commonly used: There are several autonomous computational entities, each of which has its own local memory. The entities communicate with each other by message passing. In this article, the computational entities are called computers or nodes. A distributed system may have a common goal, such as solving a large computational problem. Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users. Other typical properties of distributed systems include the following: The system has to tolerate failures in individual computers. The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program. Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.

LITERATURE SURVEY

DATA UNIDENTIFICATION:

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 unidentified 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 unidentified records can be associated, matched, and/or conjoined with other unidentified records. Data unidentification 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-unidentification.

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 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.

MAP REDUCE:

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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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 pet 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 mapped 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 unidentification 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 datasets, achieving high efficiency. However, the above approaches aim at multidimensional generalization, there by failing to work in the TDS approach. Fung etal. 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

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

multiple parties. Jiang and Clifton and Mohammed et al. proposed distributed algorithms to unidentified vertically partitioned data from different data sources without disclosing privacy information from one party to another. Jurczyk and Xiong and Mohammed et al. proposed distributed algorithms to unidentified horizontally partitioned data sets retained by multiple holders. However, the above distributed algorithms mainly aim at securely integrating and unidentifing multiple data sources. Our research mainly focuses on the scalability issue of TDS unidentification, and is, therefore, orthogonal and complementary to them. As to Map Reduce–relevant privacy protection, Roy et al. investigated the data privacy problem caused by Map Reduce and presented a system named Air vat incorporating mandatory access control with differential privacy. Further, Zhang et al. leveraged Map Reduce to automatically partition a computing job in terms of data security levels, protecting data privacy in hybrid cloud. Our research exploitsMapReduce itself to unidentified large-scale data sets before data are further processed by other Map Reduce 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 unidentification. The TDS approach, offering a good trade-off between data utility and data consistency, is widely applied for data unidentification. 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 unidentification of data sets from multiple parties, rather than the scalability aspect.

DRAWBACKS:-

- The Map Reduce computation paradigm is still a challenge to design proper Map Reduce jobs for TDS.
- The overall performance of the privacy provided is low.
- It is only suitable for the small amount of data sets.
- The unidentification of the each level is low.

PROBLEM DEFINITION

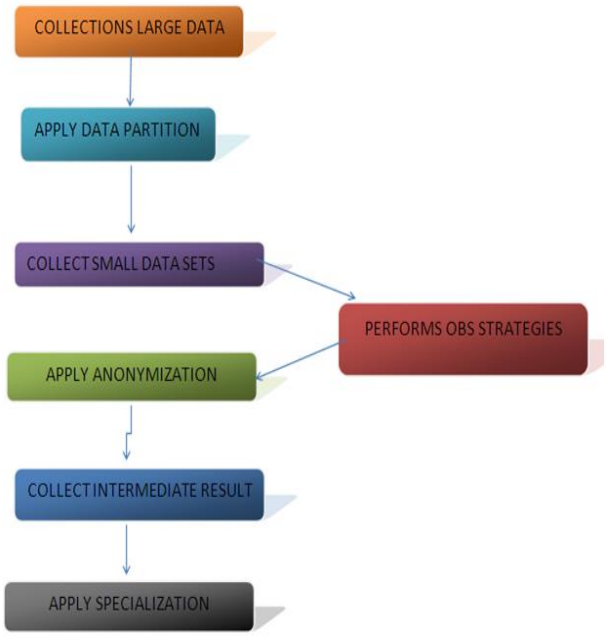
- ✓ In this paper, we propose a scalable two-phase top-down specialization (TDS) approach to unidentified large-scale data sets using the Map Reduce framework on cloud.
- ✓ In both phases of our approach, we deliberately design a group of innovative Map Reduce jobs to concretely accomplish the specialization computation in a highly scalable way.
- ✓ This approach gets input data's and split into the small data sets. Then we apply the UNIDENTIFICATION on small data sets to get intermediate result.
- ✓ Then small data sets are merged and again apply the UNIDENTIFICATION.
- ✓ We analyze the each and every data set sensitive field and give priority for this sensitive field. Then we apply UNIDENTIFICATION 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 unidentification over existing approaches.
- The overall performance of the providing privacy is high.
- Its ability to handles the large amount of data sets.
- The unidentification is effective to provide the privacy on data sets.
- Here we using the scheduling strategies to handle the high amount of datasets.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

BLOCK DIAGRAM:-



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 provide the random no for each data set.

UNIDENTIFICATION:

- ✓ After getting the individual data sets we apply the unidentification.
- ✓ The unidentification 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 unidentification levels are merged into one in the second phase. The merging of unidentification levels is completed by merging cuts. To ensure that the merged intermediate unidentification level ALI never violates privacy requirements, the more general one is selected as the merged one

MERGING:

- ✓ The intermediate results 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 unidentification called specialization.

SPECIALIZATION:

- ✓ After getting the intermediate result those results are merged into one.
- ✓ Then we again applies the unidentification 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 unidentification on specified time.
- ✓ The OBS approach is to provide the high ability on handles the large data sets.

CONCLUSION

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 provides the privacy by effective unidentification approaches. In the future work is to reduce the handling effect of large amount of the data sets. It implements the optimized balancing scheduling. Where it'sdepends on the time and size of the data sets. In this paper it have investigated the scalability problem of large-scale data unidentification by Top-Down Specialization and proposed a highly scalable two-phase TDS approach using Map Reduce

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

on cloud. Datasets are partitioned and unidentified in parallel in the first phase producing intermediate results. Then, the intermediate results are merged and further unidentified to produce consistent k-anonymous data sets in the second phase. It have creatively applied Map Reduce on cloud to data unidentification 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 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 gets input data's and split into the small data sets. Then we apply the UNIDENTIFICATION on small data sets to get intermediate result. Then small data sets are merged and again apply the UNIDENTIFICATION. Here the drawback of proposed system is there is no priority for applying the UNIDENTIFICATION on datasets. So that it takes more time to UNIDENTIFIED the datasets. So we introduce the scheduling mechanism called OPTIMIZED BALANCED SCHEDULING(OBS) to apply the UNIDENTIFICATION. 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 UNIDENTIFICATION 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 Symp. Principles of Database Systems (PODS '12), pp. 1-4, 2012.

[2] M. Armbrust, A. Fox, R. Griffith, A.D. 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[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. Lefebvre, 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. Lefebvre, D.J. DeWitt, and R. Ramakrishnan, "Mondrian Multidimensional 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. Lefebvre, 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.