Optimized Scheduling for Better Data Anonymization in Cloud using Top Down Specialization

C. Sakthi[1]
PG Scholar,
Department of Computer Science and Engineering,
J.K.K. Nattraja College of Engineering and Technology, Kumarapalayam

C. Vairavel[2]
Assistant Professor,
Department of Computer Science and Engineering,
J.K.K. Nattraja College of Engineering and Technology, Kumarapalayam

***Abstract*- Cloud computing is a way of providing on demand network access in the shared data computing resources which are accessed by millions of users. The main concern of cloud computing is provide the privacy and security issues for the cloud resource user who share their data in the public cloud. The previous workers developed some of the models for providing privacy and security like k-anonymity, l-diversity, t-closeness. In the previous work two phase top down specialization is developed where the individuals can publish their data without revealing sensitive information about them. Data Anonymization is a method that makes data worthless to anyone except the owner of the data. In big data scalability is the big issue for maintenance. However the previous work doesn't concentrate about the scheduling algorithm for balancing the load. In this work genetic algorithm is introduced for optimized scheduling algorithm.**

Keywords—: Map reduce, Big data ,Cloud computing, Resource provisioning, Performance modeling

I. INTRODUCTION

A. Introduction

BIG data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, curation, storage, search, sharing, transfer, analysis, and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, determine quality of research, prevent
diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions.

Scientists regularly encounter limitations due to large data sets in many areas, including meteorology, genomics, connect omics, complex physics simulations, and biological and environmental research. The limitations also affect Internet search, finance and business informatics. Data sets grow in size in part because they are increasingly being gathered by ubiquitous information-sensing mobile devices, aerial sensory technologies (remote sensing), software logs, cameras, microphones, radio-frequency identification readers, and wireless sensor networks. Big data is difficult to work with using most relational database management systems and desktop statistics and visualization packages, requiring instead "massively parallel software running on tens, hundreds, or even thousands of servers". What is considered "big data" varies depending on the capabilities of the organization managing the set, and on the capabilities of the applications that are traditionally used to process and analyze the data set in its domain. "For some organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data size becomes a significant consideration.

B. Overview

Big data (also spelled Big Data) is a general term used to describe the voluminous amount of unstructured and semi-structured data a company creates -- data that would take too much time and cost too much money to load into a relational database for analysis. Although Big data doesn't refer to any specific quantity, the term is often used when speaking about petabytes and exa bytes of data. A primary goal for looking at big data is to discover repeatable business patterns. It's generally accepted that unstructured data, most of it located in text files, accounts for at least 80% of an organization's data. If left unmanaged, the sheer volume of unstructured data that's generated each year within an enterprise can be costly in terms of storage. Unmanaged data can also pose a liability if information cannot be located in the event of a compliance audit or lawsuit. Big data analytics is often associated with cloud computing because the analysis of large data sets in real-time requires a framework like Map Reduce to distribute the work among tens, hundreds or even thousands of computers.

C. Hadoop File System

Apache Hadoop is an open-source software framework for storage and large scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a

global community of contributors and users. It is licensed under the Apache License 2.0.
The Apache Hadoop framework is composed of the following modules :

- Hadoop Common - contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS) - a distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster.
- Hadoop YARN - a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.
- Hadoop Map Reduce - a programming model for large scale data processing.

All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are common and thus should be automatically handled in software by the framework. Apache Hadoop's Map Reduce and HDFS components originally derived respectively from Google's Map Reduce and Google File System (GFS) papers

Hadoop consists of the Hadoop Common package, which provides files system and OS level abstractions, a Map Reduce engine and the Hadoop Distributed File System (HDFS). The Hadoop Common package contains the necessary Java Archive (JAR) files and scripts needed to start Hadoop. The package also provides source code, documentation and a contribution section that includes projects from the Hadoop Community.

For effective scheduling of work, every Hadoop-compatible file system should provide location awareness: the name of the rack where a worker node is. Hadoop applications can use this information to run work on the node where the data is, and, failing that, on the same rack/switch, reducing backbone traffic. HDFS uses this method when replicating data to try to keep different copies of the data on different racks. The goal is to reduce the impact of a rack power outage or switch failure, so that even if these events occur, the data may still be readable. A small Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a Job Tracker, Task Tracker, Name Node and Data Node. A slave or worker node acts as both a Data Node and Task Tracker, though it is possible to have data-only worker nodes and compute-only worker nodes. These are normally used only in nonstandard applications. Hadoop requires Java Runtime Environment (JRE) 1.6 or higher. The standard start-up and shutdown scripts require Secure Shell(ssh) to be set up between nodes in the cluster.

D. Applications

The HDFS file system is not restricted to Map Reduce jobs. It can be used for other applications, many of which are under development at Apache. The list includes the HBase database, the Apache Mahout machine learning system, and the Apache Hive Data Warehouse system. Hadoop can in theory be used for any sort of work that is batch-oriented rather than real-time, that is very data-intensive, and able to work on pieces of the data in parallel. As of October 2009, commercial applications of Hadoop included:

- Log and/or clickstream analysis of various kinds
- Marketing analytics
- Machine learning and/or sophisticated data mining
- Image processing
- Processing of XML messages
- Web crawling and/or text processing

General archiving, including of relational/tabular data, e.g. for compliance

With the deployment of web applications, scientific computing, and sensor networks, a large amount of data can be collected from users, applications, and the environment. For example, user click through data has been an important data source for improving web search relevance and for understanding online user behaviors. Such datasets can be easily in terabyte scale; they are also continuously produced. Thus, an urgent task is to efficiently analyze these large datasets so that the important information in the data can be promptly captured and understood. As a flexible and scalable parallel programming and processing model, recently Map Reduce (and its open source implementation Hadoop) has been widely used for processing and analyzing such large scale datasets. On the other hand, data analysts in most companies, research institutes, and government agencies have no luxury to access large private Hadoop/Map Reduce clouds.

Therefore, running Hadoop/Map Reduce on top of a public cloud has become a realistic option for most users. In view of this requirement, Amazon has developed Elastic Map Reduce that runs on-demand Hadoop/Map Reduce clusters on top of Amazon EC2 nodes. There are also scripts1 for users to manually setup Hadoop/Map Reduce on EC2 nodes.

Computing devices have numerous uses and are essential for businesses, scientists, governments, engineers, and the everyday consumer. What all these devices have in common is the potential to generate data. Essentially, data can come from anywhere. Sensors gathering climate data, a person posting to a social media site, or a cell phone GPS signal are example sources of data. The popularity of the Internet alongside a sharp increase in the network bandwidth available to users has resulted in the generation of huge amounts of data. In response to these very same issues, engineers at Google developed the Google File System, a distributed file system architecture model for large-scale data processing and created the Map Reduce programming model.

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 and a Reduce() procedure that performs a summary operation. The Map Reduce System 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 or a grid.

Hadoop is an open source software implementation of Map Reduce, written in Java. Hadoop was created in response to the need for a Map Reduce framework that was unencumbered by proprietal licenses, as well as the growing need for the technology in Cloud computing. It focuses on Hadoop and investigates the load balancing mechanism in Hadoop's Map Reduce framework for small- to medium-sized clusters. This is an important area of research for several reasons. In addition to that to provide effective scheduling it use Map Reduce Task Scheduling algorithm for Deadline constraints. In this technique a node classification algorithm is used by measuring the node's computing capacity. Through this distribution of the data according to the node capacity. In addition to that it use micro-partioning methods to break the workload into many small tasks that are dynamically scheduled at runtime. This approach is only effective in systems with high-throughput, low-latency task schedulers and efficient data materialization.

II. LITERATURE SURVEY

A Scalable Two-Phase Top Down Approach Specialization For Data Anonymization Using MapReduce On Cloud- Xuyun Zhang, Chang Liu, Laurence T Yung,JinJun Chen

A large number of cloud services require users to share private data like electronic health records for data analysis or mining, bringing privacy concerns. Anonymizing data sets via generalization to satisfy certain privacy requirements such as k-anonymity is a widely used category of privacy preserving techniques. At present, the scale of data in many cloud applications increases 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, it is a challenge for existing anonymization approaches to achieve privacy preservation on privacy-sensitive large-scale data sets due to their insufficiency of scalability. In this paper, we propose a scalable two-phase top-down specialization (TDS) approach to anonymize large-scale data sets using the MapReduce framework on cloud.. Experimental evaluation results demonstrate that with our approach, the scalability and efficiency of TDS can be significantly improved over existing approaches

B Security and Privacy Security and Privacy Computing Environments – Hassan Takabi and James B.D. Joshi, Gail-Joon Ahn – 2010

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three delivery models, and four deployment models. Cloud computing aims to consolidate the economic utility model with the evolutionary development of many existing approaches and computing technologies, including distributed services, applications, and in-formation infrastructures consisting of pools of computers, networks, and storage resources. It provides access to data, but the challenge is to ensure that only authorized entities can gain access to it. It can use different security, privacy, and trust requirements and potentially employ various mechanisms, interfaces, and semantics. Such domains could represent individually enabled services or other infrastructural or application components.

C A Privacy Leakage Upper Bound Constraint-Based Approach for Cost-Effective Privacy Preserving of Intermediate Data Sets in Cloud - Xuyun Zhang, Chang Liu, Surya Nepal, Suraj Pandey, and Jinjun Chen

Cloud computing is regarded as an ingenious combination of a series of technologies, establishing a novel business model by offering IT services and using economies of scale. Participants in the business chain of cloud computing can benefit from this novel model. Cloud customers can save huge capital investment of IT infrastructure, and concentrate on their own core business. Preserving the privacy of intermediate data sets becomes a challenging problem because adversaries may recover privacy-sensitive information by analysing multiple intermediate data sets. In this work Privacy Leakage Upper Bound Constraint-Based Approach is proposed to provide a privacy concern for the data sets in the cloud. It work based on identifying the intermediate data sets to be encrypted iteratively. A tree structure has been modelled from the generation relationships of intermediate data sets to analyse privacy propagation among data sets.

III. PROBLEM DEFINITION

In the case of handling large amount of data (big data) privacy is the main issue where the identity of users has to be in hidden. When applying specialization technique in partitioned data's, the optimized feature set have to be find out in order to apply specialization efficiently. The optimized feature set can be find out by using Genetic technique. Optimized feature set in the sense that which kind of attributes has to be specialized

IV. EXISTING SYSTEM

In Existing work two phase top down specialization approach is implemented. This method is used to provide data anonymization for the user by hiding their identity information. This is achieved by specialization technique which is used to hide the identity of the users by Information Gain Privacy Loss (IGPL) metric. This is specialization is applied in two phase level where given entire data is divided into number of partitions.

In the first phase entire data is divided into multiple partitions by using mapReduce problem where partitions are independent to each other. After dividing data into partitions, the specialization technique is applied to each and every partition individually. In the second phase, all specialized partitions are merged together. Finally the specialization will apply to the merged data. This is continued until achieves the k-anonymity level.

V. PROPOSED SYSTEM

In the computer science field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.
Genetic algorithms find application in bioinformatics, phylogenetics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics, pharmacometrics and other fields.

Anonymization level can intuitively represent the anonymization degree of a data set, i.e., the more specific AL a data set has, the less degree of anonymization it corresponds to. Thus, TDS approaches employ anonymization level to track and manage the specialization process. Thus the Genetic can be applied to find the best anonymization level in order to achieve the better specialization.

machines in order to produce smaller tasks. These tasks are assigned to reduce machines in a “just-in-time” fashion as workers become idle, allowing the task scheduler to dynamically mitigate skew and stragglers. Running many small tasks lessens the impact of stragglers, since work that would have been scheduled on slow nodes when using coarse-grained tasks can now be performed by other idle workers. With large tasks, it can be more efficient to exclude slow nodes rather than assigning them any work. By assigning smaller units of work, jobs can derive benefit from slower nodes.

Advantages

- Best optimized strategy by selecting the appropriate feature set.
- Data anonymization is achieved in k- anonymity level.

VI. EXPERIMENTS

A. *Hadoop Setup:*

In this module, hadoop installation will be done. In our work hadoop is installed on the windows OS and initial configurations are done to support the processing of specialization. Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation. After installation and configuration of hadoop, the input data which is to be specialized to hide the user information will be gathered from the user. The input data will be associated with the hadoop environment which is to be evaluated and processed in differenct anonymization level and finally completely anonymized output data will be retrieved.

B. Data Partitioning Using Mapper

When D is partitioned into D_i, 1 <= I <= p, it is required that the distribution of data records in D_i is similar to D.A data record here can be treated as a point in an m-dimension space, where m is the number of attributes. Thus, the intermediate anonymization levels derived from D_i, 1 <= i <= p, can be more similar so that we can get a better merged anonymization level. Random sampling technique is adopted to partition D, which can satisfy the above requirement. Specifically, a random number rand, 1 <= rand <= p, is generated for each data record. A record is assigned to the partition D_{rand}.

Algorithm:
Input: Data record (IDr, r), r 2 D, partition parameter p.
Output: Di, 1 _ i _ p.
Map: Generate a random number rand,
where 1 <= rand <= p;
emit (rand, r).
Reduce: For each rand,
emit (null, list(r))..

C. Data Mapping Using Genetic Algorithm

In this module, the partioned data’s will be allocated to the VM in which best optimized anonymization level can be achieved. The optimized allocation of resources is done in our work by using the Genetic method which is an bio-logically inspired approach. In genetic, each and every possible allocation of data partition into VM will be considered as an gene. The best optimized output is selected by using the following algorithm.

Algorithm:
[Start] Generate random population of *n* chromosomes (suitable solutions for the problem)

[Fitness] Evaluate the fitness *f(x)* of each chromosome *x* in the population
[New population] Create a new population by repeating following steps until the new population is complete
[Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
[Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
[Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
[Accepting] Place new offspring in a new population
[Replace] Use new generated population for a further run of algorithm
[Test] If the end condition is satisfied, **stop**, and return the best solution in current population
[Loop] Go to step **2**

D. Data Anonymization Using TDS

In module, data anonymization is done by using the top down specialization approach. TDS is an iterative process starting from the top most domain values in the taxonomy trees of attributes. Each round of iteration consists of three main steps, namely, finding the best specialization, performing specialization and updating values of the search metric for the next round. Such a process is repeated until k-anonymity is violated, to expose the maximum data utility. The goodness of a specialization is measured by a search metric. We adopt the information gain per privacy loss (IGPL), a tradeoff metric that considers both the privacy and information requirements, as the search metric in our approach. A specialization with the highest IGPL value is regarded as the best one and selected in each round. Given a specialization spec: p → Child (p), the IGPL of the specialization is calculated byIGPL (spec) = IG (spec) / (PL (spec) + 1)
The term IG (spec) is the information gain after performing spec, and PL (spec) is the privacy loss. IG (spec) and PL (spec) can be computed via statistical information derieved from data sets. Let Rx denotes the set of original records containing attributes values that can be generalized to x. |Rx| is the number of data records in Rx. Let I(Rx) be the entrophy of Rx. Then IG (spec) is calculated by

$$IG(spec) = I(Rp) - \sum_{c \in Child(p)} \left(\frac{|R_c|}{|R_p|} \right) I(R_c)$$

Let |(Rx , sv)| denote the number of the data records with sensitive value sv in Rx.I(Rx) is computed by

$$I(Rx) = -\sum_{sv \in SV} \left(\frac{|(R_x, S_v)|}{|R_x|} \right) . \log_2 \frac{|(R_x, S_v)|}{|R_x|}$$

The anonymity of a data set is defined by the minimum group size out of all QI group, denoted as A.

E. Data reduction using Reducer

All intermediate anonymization levels are merged into one in the second phase. The merging of anonymization levels is completed by merging cuts. Specifically, let Cuta in AL0 a and Cutb in AL0 b be two cuts of an attribute. There exist domain values qa 2 Cuta and qb 2 Cutb that satisfy one of the three conditions: qa is identical to qb, qa is more general than qb, or qa is more specific than qb. To ensure that the merged intermediate anonymization level ALI never violates privacy requirements, the more general one is selected as the merged one, for example, qa will be selected if qa is more general than or identical to qb. For the case of multiple anonymization levels, we can merge them in the same way iteratively. The following lemma ensures that ALI still complies privacy requirements.

E. Performance Evaluation

In this module, the performance evaluation is done compare the efficiency of our proposed work with the existing work. The comparison of proposed work with the existing work is done by using some performance metrics called accuracy and anonymization level

VII. CONCLUSION

In this work TDS approach is used to handle the partitioned data sets effectively. This approach gives effective way of preserving privacy information of the user by hiding user's sensitive information. Genetic technique is used to find out the generalized feature set for efficient specialization. It reduces the computation cost considerably by selecting the optimal feature set for specialization. Delay can be also reduced. The specialization is applied just for optimal feature set, instead of applying specialization for entire data. So that optimized data anonymization is also achieved.

REFERENCES

1. Hassan Takabi and James B.D. Joshi, Gail-Joon Ahn, "Security and Privacy Security and Privacy Computing Environments", *Copublished By The Ieee Computer And Reliability Societie,* PP. 24-31, 2010.
2. Xuyun Zhang, Chang Liu, Surya Nepal, Suraj Pandey, and Jinjun Chen, "A Privacy Leakage Upper Bound Constraint-Based Approach for Cost-Effective Privacy Preserving of Intermediate Data Sets in Cloud", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 24, No. 6, June 2013.
3. Hsiao-Ying Lin, and Wen-Guey Tzeng, "A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 23, No. 6, June 2012.
4. Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data*", IEEE Transactions On Parallel And Distributed Systems*, Vol. 25, No. 1, January 2014.
5. Ninghui Li, Tiancheng Li, and Suresh Venkata Subramanian, "Closeness: A New Privacy Measure for Data Publishing*", IEEE Transactions On Knowledge And Data Engineering*, Vol. 22, No. 7, July 2010.
6. Tiancheng Li, Ninghui Li, Jian Zhang, and Ian Molloy, "Slicing: A New Approach for Privacy Preserving Data Publishing", *IEEE Transactions On Knowledge And Data Engineering*, Vol. 24, No. 3, March 2012.
7. Davide Bacciu, Alessio Micheli, and Alessandro Sperduti, "Compositional Generative Mapping for Tree-Structured Data—Part I: Bottom-Up Probabilistic Modeling of Trees*", IEEE Transactions*

On Neural Networks And Learning Systems, Vol. 23, No. 12, December 2012.

8. Dimitrios Zissis, Dimitrios Lekkas, "Addressing cloud computing security issues*", Future Generation Computer Systems,* 28 PP. 583–592, 2012.
9. Kehuan Zhang, Xiaoyong Zhou, Yangyi Chen and XiaoFeng Wang, and Yaoping Ruan, "Sedic: Privacy-Aware Data Intensive Computing on Hybrid Clouds", *Proc. 18th ACM Conf. Computer and Comm. Security (CCS '11),* pp. 515-526, 2011.
10. Noman Mohammed, Benjamin C. M. Fung, and Mourad Debbabi, "Anonymity meets game theory: secure data integration with malicious participants", *The VLDB Journal*, 20:567–588, 2011