INTERNATIONAL JOURNAL OF CURRENT MEDICAL AND PHARMACEUTICAL RESEARCH

IJCMPR

Available Online at http://www.journalcmpr.com

RESEARCH ARTICLE

MEDICAL APPLICATION OF PRIVACY PRESERVATION BY BIG DATA ANALYSIS USING HADOOP MAP REDUCE FRAMEWORK

Ashish P*, Tejas S, Srinivasa J G and Sumeet G

Department of ISE, DSCE, Bangalore

ARTICLE INFO

Article History:

Received 15 August, 2015
Received in revised form
30 August, 2015
Accepted 20 September, 2015
Published online 28 September, 2015

Key words:

Annonymisation, Big data analytics, Health care, Map reduce, Cloud computing, Privacy preservation.

ABSTRACT

The Map Reduce framework has become the de-facto framework for large-scale data analysis and data mining. The computer industry is being challenged to develop methods and techniques for affordable data processing on large datasets at optimum response times. The technical challenges in dealing with the increasing demand to handle vast quantities of data is daunting and on the rise. One of the recent processing models with a more efficient and intuitive solution to rapidly process large amount of data in parallel is called Map Reduce. It is a framework defining a template approach of programming to perform large-scale data computation on clusters of machines in a cloud computing environment. Map Reduce provides automatic parallelization and distribution of computation based on several processors. It hides the complexity of writing parallel and distributed programming code

INTRODUCTION

Healthcare is one of the most important areas for developing and developed countries to facilitate the priceless human resource. Nowadays healthcare industry is flooded with enormous amount of data that need validation and accurate analysis. Even though Big Data Analytics and Hadoop can contribute a major role in processing and analyzing the healthcare data in variety of forms to deliver suitable applications, India takes the second place in the world in its population. Increasing population in India over-burdens the health care structure in the country. The exponential growth of data over the last decade has introduced a new domain in the field of information technology called Big Data. Here we propose map reduce to analyze medical big data of India, Map Reduce is a new framework specifically designed for processing huge datasets on distributed sources.

Apache's Hadoop is an implementation of Map Reduce. Currently Hadoop has been applied successfully for file based datasets. The execution engine that is developed on top of Hadoop applies Map and Reduce techniques to break down the parsing and execution stages for parallel and distributed processing. Moreover Map Reduce will provide the fault tolerance, scalability and reliability because its library is designed to help process very large amount of data using hundred and thousands of machine, which must tolerate the machine failure.

Map Reduce

Map Reduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

Our implementation of Map Reduce runs on a large cluster of commodity machines and is highly scalable: a typical Map Reduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of Map Reduce programs have been implemented and upwards of one thousand Map Reduce jobs are executed on Google's clusters every day. Workers and assigns each one a map or a reduce task according to the stage. Before starting the Map task, an input file is loaded on the distributed file system. At loading, the file is partitioned into

multiple data blocks which have the same size, typically 64MB, and each block is triplicate to guarantee fault-tolerance. Each block is then assigned to a mappers, a worker which is assigned a map task, and the mapper applies Map () to each record in the data block. The intermediate outputs produced by the mappers are then sorted locally for grouping key value pairs sharing the same key. After local sort, Combine () is optionally applied to perform pre aggregation on the grouped key-value pairs so that the communication cost taken to transfer all the intermediate outputs to reducers is minimized. Then the mapped outputs are stored in local disks of the mappers, partitioned into R, where R is the number of Reduce tasks in the MR Job. This partitioning is basically done by a hash function e.g., hash (key) mod R. When all Map tasks are completed, the Map Reduce scheduler assigns Reduce tasks to workers. The intermediate results are shuffled and assigned to reducers via HTTPS protocol. Since all mapped outputs are already partitioned and stored in local disks, each reducer performs the shuffling by simply pulling its partition of the mapped outputs from mappers. Basically, each record of the mapped outputs is assigned to only a single reducer by one-to-one shuffling strategy. Note that this data transfer is performed by reducers' pulling intermediate results. A reducer reads the intermediate results and merges them by the intermediate keys, i.e. key2, so that all values of the same key are grouped together. This grouping is done by external merge-sort. Then each reducer applies Reduce () to the intermediate values for each key2 it encounters. The output of reducers is stored and triplicate in HDFS.

Note that the number of Map tasks does not depend on the number of nodes, but the number of input blocks. Each block is assigned to a single Map task. However, all Map tasks do not need to be executed simultaneously and neither are Reduce tasks. For example, if an input is broken down into 400 blocks and there are 40 mappers in a cluster, the number of map tasks is 400 and the map tasks are executed through 10 waves of task runs. The Map Reduce framework executes its tasks based on runtime scheduling scheme. It means that Map Reduce does not build any execution plan that specifies which tasks will run on which nodes before execution. While DBMS generates a query plan tree for execution, a plan for executions in Map Reduce is determined entirely at runtime. With the runtime scheduling, Map Reduce achieves fault tolerance by detecting failures and reassigning tasks of failed nodes to other healthy nodes in the cluster. Nodes which have completed their tasks are assigned another input block. This scheme naturally achieves load balancing in that faster nodes will process more input chunks and slower nodes process less input in the next

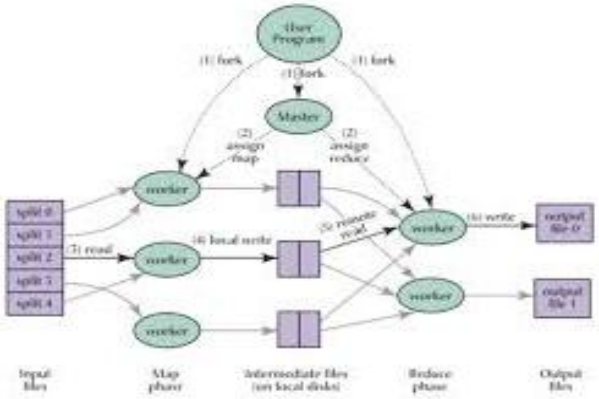

Figure 1 Map Reduce architecture

wave of execution. Furthermore, Map Reduce scheduler utilizes a speculative and redundant execution. Tasks on straggling nodes are redundantly executed on other idle nodes that have finished their assigned tasks, although the tasks are not guaranteed to end earlier on the new assigned nodes than on the straggling nodes. Map and Reduce tasks are executed with no communication between other tasks. Thus, there is no contention arisen by synchronization and no communication cost between tasks during a MR Job execution

Related Work

Healthcare In India

D. Peter Augustine [1] Despite the government has promised to introduce digitization for maintaining medical records, the reality is not as expected. The country does not even have standardization in common medical terminologies. In big data processing the data must be process in a distributed environment. The requirement for analyzing data such as medical information requires statistical and mining approach for analyzing the data. Delivering the data in a faster response time will be at higher priority.

Mapreduce In Cloud Computing

Samira Daneshyar and Majid Razmjoo[2] This work is to proposes a framework for running Map Reduce system in a cloud environment based on the captured requirements and to present its implementation on Amazon Web Services. They also presents an experimentation of running the Map Reduce system in a cloud environment to validate the proposed framework and to present the evaluation of the experiment based on the criteria such as speed of processing, data-storage usage, response time and cost efficiency

Analysis Of Bidgata Using Apache Hadoop And Map Reduce

Mrigank Mridul *et al*. [3] Map educe is a software framework introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers. Map Reduce is a programming model for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs and a reduce function that merges all intermediate values associated with the same intermediate key.

Top-Down Specialization For Information And Privacy Preservation

Benjamin *et al*. proposed the privacy goal is specified by the anonymity on a combination of attributes called a virtual identifier, where each description on a virtual identifier is required to be shared by some minimum number of records in the table. A generalization taxonomy tree is specified for each categorical attribute in a virtual identifier. We present a Top-Down Specialization (TDS) approach to generalize a table to satisfy the anonymity requirement while preserving its usefulness to classification. TDS generalizes the table by specializing it iteratively starting from the most general state.

Parallel Data Processing With Mapreduce: A Survey

Aman Kansal *et al*. [5] describe the challenges developers face

in optimizing software for energy efficiency by exploiting application-level knowledge. To address these challenges, we propose the development of automated tools that profile the energy usage of various resource components used by an application and guide the design choices accordingly. Henri Arjamaa *et al*. [6] present energy consumption estimates of ICT equipment in Finland and in three important industrial countries, namely the United States, Germany, and the United Kingdom. In addition, a worldwide estimate of the energy consumption of data centers is presented. The results are then analyzed, which give answers to questions, such as how valid are the estimation methods used and are the estimation methods comparable with each other. Christopher K. Lennard *et al*. [7] describe resynthesis procedures used for reducing power consumption in CMOS networks have produced poor results as they select nodes for resynthesis based upon local circuit properties. In this, a technique is presented for optimizing the choice of regions used in resynthesis. The cost function which is developed is able to predict the amount of global improvement in power expected through the resynthesis of network nodes under both zero as well as arbitrary delay assumptions. Pinheiro *et al*. [8] have proposed a technique for managing a cluster of physical machines with the objective of minimizing the power consumption, while providing the required Quality of Service (QoS). The authors use the throughput and execution time of applications as constraints for ensuring the QoS. Here nodes are assumed to be homogeneous. The algorithm periodically monitors the load and decides which nodes should be turned on or off to minimize the power consumption by the system, while providing expected performance.

Srikantaiah *et al*. [9] have investigated the problem of dynamic consolidation of applications in virtualized heterogeneous systems in order to minimize energy consumption, while meeting performance requirements. The authors have explored the impact of the workload consolidation on the energy-per-application metric depending on both CPU and disk utilizations. Elnozahy *et al*. [10] have investigated the problem of power-efficient resource management in a single web-application environment with fixed response time and load-balancing handled by the application. The two main power-saving techniques are switching power of computing nodes on or off and Dynamic Voltage and Frequency Scaling (DVFS). Nathuji and Schwan [11] have studied power management techniques in the context of virtualized data centers, which has not been done before. Besides hardware scaling and VMs consolidation, the authors have introduced and applied a new power management technique called "soft resource scaling. Dodonov *et al*. [12] have proposed an approach to scheduling distributed applications in Grids.

Improving Mapreduce Performance In Heterogeneous Environments

Matei Zaharia *et al*. -describes the sheer volume of data that these services work with has led to interest in parallel processing on commodity clusters. The leading example is Google, which uses its Map Reduce framework to process 20 petabytes of data per day [1]. Other Internet services, such as ecommerce websites and social networks, also cope with enormous volumes of data. These services generate click stream data from millions of users every day, which is a potential gold mine for understanding access patterns and increasing ad revenue. Furthermore, for each user action, a web application generates one or two orders of magnitude more data in system logs, which are the main resource that developers and operators have for diagnosing problems in production.

Direct Qr Factorizations For Talland-Skinny Matrices In Mapreduce Architectures

Austin R. Benson *et al*. describes two fundamental matrix decompositions with applications throughout scientific computing and data analysis. They are The QR factorization and the SVD. Matrices with many more rows than columns, so called "tall-and-skinny matrices," there is a numerically stable, efficient, communication-avoiding algorithm for computing the QR factorization. It has been used in Traditional high performance computing and grid computing environments

A Study On Authorized Deduplication Techniques In Cloud Computing

Bhushan Choudhary and Amit Dravid, describes the new emerging trends in the new generation technology and the amount of data increasing to hundreds of Petabytes and about data Deduplication. Data deduplication is the technique which compresses the data by removing the duplicate copies of identical data and it is extensively used in cloud storage to save bandwidth and minimize the storage space. To secure the confidentiality of sensitive data during deduplication, the convergent encryption technique is used to encrypt the data before outsourcing. Data deduplication was proposed to secure the information security by counting differential benefits of clients in the copy check. Security examination shows that the plans are secure as far as insider and outsider attacks determined in the proposed security model.

The Family Of Mapreduce And Large Scale Data Processing Systems

Sherif Sakr *et al*. describes the provides a comprehensive survey for a family of approaches and mechanisms of large scale data processing mechanisms that have been implemented based on the original idea of the Map Reduce framework and are currently gaining a lot of momentum in both research and industrial communities. Several large scale data processing systems that resemble some of the ideas of the Map Reduce framework for different purposes and application scenarios.

Map-Based Graph Analysis On Mapreduce

Upa Gupta *et al*. describes the a new design pattern for a family of iterative graph algorithms for the MapReduce framework. Each MapReduce node participating in the graph analysis task reads the same graph partition at each iteration step, which is made local to the node, but it also reads all the current analysis results from the distributed file system (DFS). The algorithm requires one Map Reduce job for pre-processing the graph and the repetition of one map-based MapReduce job for the actual analysis.

Anonymizing Classification Data For Privacy Preservation

Benjamin C.M, *et al*. describes the a useful approach to combat such linking attacks, called kanonymization, is

anonymizing the linking attributes so that at least k released records match each value combination of the linking attributes. Minimizing the distortion to the training data is not relevant to the classification goal that requires extracting the structure of predication on the "future" data. The author mainly proposes a k-Annonymisation solution for classification. To find a k-Annonymisation, not necessarily optimal in the sense of minimizing data distortion, which preserves the classification structure? Experiments on real-life data show that the quality of classification can be preserved even for highly restrictive anonymity requirement.

Existing System

The k-anonymity, l-diversity, t-closeness, each of them have several drawbacks as indicated it includes data utility loss and sensitivity disclosure. To overcome this author in has indicated a method personalized privacy preservation which takes in to account record owners privacy requirement. In the record owner can indicate his privacy by indicating in terms of a guarding node. The values of it are based on a sensitive hierarchy which is framed by Data Publisher. The core of this technique is to divide the sensitive value based on importance so that more privacy is given to those values and data utility is improved. The drawback of this method is that it may require several iterations based on the guarding node, sensitive attribute is also generalized which has larger information loss. The most important drawback is that distribution of sensitive attribute has not been taken in to account while anonymization.

Proposed System

In this project we propose a novel privacy preserving technique that over comes the disadvantages and other anonymization techniques. The core of this method can be divided in to two major components. The first component deals with additional attribute used in the table which is in the form of flags. Sensitive Disclosure Flag (SDF) determines whether record owner sensitive information is to be disclosed or whether privacy should be maintained. The second flag that we are using is Sensitive Weigh (SW) which indicates how much sensitive the attribute is. SDF is dependent on SW. Second component deals with a novel representation called Frequency Distribution Block (FDB) and Quasi–Identifier Distribution Block (QIDB) used for measuring the distribution.

Block Diagram

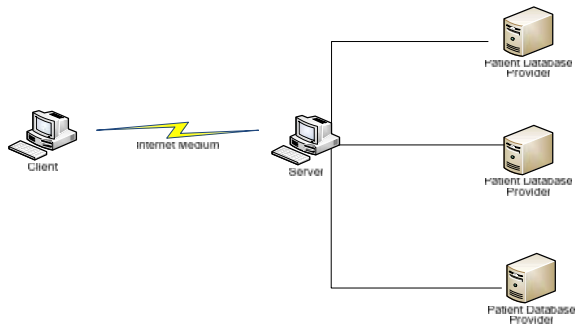

Figure 2 Block Diagram of Proposed System

The working of the projects involves the distribution of patients on multiple hospital servers' and being centralized via a server.

Our approach provides protections of data from hackers the working is as follows

Objective of our approach is to find a generalized table T* such that distribution of each QIDB is approximately equal to the diversity of the overall distribution which is there in FDB. For simplicity the entire quasi identifiers are represented as Q and their values as q. Similarly we assume there is a single sensitive attribute S and its value is s. Relation T is made of n number of tuples T={t1,t2,…,tn}. Record owner information can be retrieved by referring as ti.s to indicate sensitive value and ti.q for quasi identifier value 1.

Requirement For Personal Privacy

Sensitive Weight

For each tuple t T, its sensitive weight is added. This value is taken from Relation W(d,sw) where d disease and SW,sensitive weight. W contains k records.

ti.sw={ wj.sw if wj.d=ti.s 1<= j <= K } $\forall$ 1 ≤ I ≤ n

Sensitive Disclosure Flag

for each tuple t T, its sensitive Disclosure Flag is indicated as t.sdf.

ti.sdf={1 if ti.sw=0 *ud ti.sw=1 }* $\forall$ 1 ≤ I ≤ n

ud represents user defined and the value is either 0 or 1. ti.sdf=0 then user is not ready to disclose his information and ti.sdf=1 then user is ready to disclose his information. In table 5 value of sw and sdf are indicated assuming that sdf value is accepted from record owner for SW=1. We can also observe that if sw=0 its correspondent sdf is initialized to 1 indicating that the sensitivity of this record is not of much relevance.

Thresholds For Personalized Privacy

Threshold values are defined for various dimensions of personalized privacy to improve the overall performance of generalization, suppression and disclosure.

1) $TH\beta_n$ minimum number of records in T.
2) $TH\beta_{iter}$ maximum number of iterations that must be performed . it indicates the amount of generalization & Height (VDH)
3) $TH\beta_{suppr}$ minimum number of sensitive values for suppression.
4) $TH\beta_{disc}$ minimum number of sensitive values for disclosure.
5) $TH\beta_{acct}$ minimum threshold that can be added or subtracted.

Since we are considering the distribution aspect we can indicate different threshold values. The first value indicates the minimum number of tuples that must be present for applying anonymization which was never defined in the previous representations. $TH\beta_{iter}$ based on the knowledge of the height of Value domain hierarchy. The larger the value of $TH\beta_{iter}$ higher the generalization and consequently information loss is more $TH\beta_{suppr}$ indicates the minimum number of sensitive distribution that may be there in QIDB for removal of that

block after $TH\beta_{iter}$. $TH\beta_{disc}$ indicates the threshold value that can be added or subtracted to each frequency distribution for each disease such that it is equivalent to the distribution FDB. The frequency of QIDB block and FDB will not be exactly same so while checking the distribution of each disease is checked whether the frequency in that *qidb.v.s*±$TH\beta_{acct}$ always $TH\beta_{disc} > TH\beta_{acct}$.

Additional Block Creations For Personal Privacy

Frequency Distribution Block

Distribution of each wj.d with respect to the original distribution ti.s is saved in relation FDB (d,p) where d indicates disease and p indicates probability distribution of it. Each p for d is calculated by mapping each d in T to the total number of tuples in T i.e. n, $\forall\ 1 \le u \le k$. let us assume there are m records in the relation.

Quasi–Identifier Distribution Block

For each ti.s where ti.sw=1 & ti.sdf=0 a new QIDB is created containing $t_1.s\ \forall\ 1 \le i \le n$.The relation QIDB.V(q,s) where $qidb.v_l.q = t_i.q$ *and* $qidb.v_l.s = t_i.s$**.** Let us assume there are dn QIDB blocks.

Functions For Personal Privacy

Generalization

A general domain of an attribute T.Q is given by a generalization function. Given a value t.q in the original domain, function returns a generalized value within the domain.

For each t we use t* to represent its generalized tuple in T*.we denote it as G (T).

This is similar to earlier representations let us assume that Domain Generalization Hierarchy and Value Generalization Hierarchy are defined for each Quasi Identifiers.

The distance vector of quasi attributes has also been generated. In figure 1 Value and Domain Generalization Hierarchy of zipcode has been indicated. Age is also generalized similarly.

(Check Frequency) for any QIDB, we check CF (QIDB.V) wither QIDB.V frequency of distribution is equal to the frequency distribution in FDB. It is done as follows

Let c be the no of records in QIDB.V. for each UNIQ(qidb.vl.s) find total no of mappings which match $qidb.v_l.s$ to the no of records i.e. c in QIDB.V, thus CF will return true if

$\forall\ 1 <= u <= m$ such that $fdb_u.d = qidb.v_l.s$
$Fdb_u.p \approx \underline{UNIQ\ (qidb.V_L.S)} \pm TH\beta_{acct}c$

This is checked in every iteration if a QIDB satisfies the frequency distribution then this block will not be considered for the next iteration.

(Suppression) After $TH\beta_{iter}$ iterations, SUPP (QIDB.v) suppress the block if it satisfies the following condition

$\forall\ 1 \le l \le c$ such that for every $fdb_u.d=qidb.v_l.s\ \wedge\ fdb_u.d=w_j.d\ \wedge\ w_j.sw=1$ for some $j\ 1 \le j \le k$

Count $(qidb.V_L.S) \le TH\beta_{suppr}$

(Disclosure) After $TH\rho_{iter}$ iterations, DIS (QIDB.v) adds additional records if it satisfies the following condition

$\forall\ 1 \le l \le c$ such that for every $fdb_u.d=qidb.v_l.s\ \wedge\ fdb_u.d=w_j.d\ \wedge\ w_j.sw=1$ for some $j\ 1 \le j \le k$

$$\frac{UNIQ(qidb.v_l.s)}{c} \approx TH\rho_{disc} \pm fdb_u.p$$

Personalized Privacy Breach

Consider an attacker who attempts to infer the sensitive data of a record owner x. the worst case scenario assumes that the adversary knows Q of X, therefore the attacker observes only those tuples $t^* \in T'$ whose Q value ti*.q covers x.q for all i such that $1 \le I \le n$. These tuples form a Q-group. That is, if ti* and tip* are two such tuples then ti*.q=tip*.q for all i such that $1 \le I \le n$ if this group is not formed the attacker cannot infer sensitive attribute of x.

Required Q-Group/ Act(X)

Given an individual x, the Required Q-group RG(X) is the only Q-group in t* covers x.q. let us assume ACT(X) refers to those records which are generalized to RG(X). ACT(X) is unknown to the attacker. To obtain ACT(X), the attacker must find some external data base EXT(X) that must be covered in RG(X).

External Data Base Ext(X)

EXT(X) are set of individuals whose value is covered by RG(X) In general ACT(X) ⊆ EXT X .The attacker adopts a combinational approach to infer sensitive attribute of x. let us assume that x.s is present in one of ti* and the repetition of x is not present. The possible reconstruction of the RG(X) includes r distinct record owners x1,x2,x3,…,xr who belong to EXT(X) are taken but there can be only y in RG(X). this can be understood by the probabilistic nature and can be indicated as perm(r,y). perm(r,y) is Possible Reconstruction(PR) that can be formed by using r owners and y mappings. Breach Probability (BP) indicates the probability of inferred knowledge. Let us assume ACTN indicates actual number of records with sensitive attribute that can be inferred to x.

$$BP = \frac{ACTN}{perm\,(r,y)}$$

BP will decide the privacy parameter, BP is 100% then x can be inferred if it is very low than the inference will be very much difficult for the attacker.

Qidb-Anonymization Algorithm

In this algorithm we are using a sequential processing of quasi values since the assumption is that in each region usually the distribution of sensitivity is approximately same. The algorithm is as follows,

Algorithm QIDB-Anonymization

Input: private data T with SW-SDF, threshold values

$THρ_n$ $THρ_{iter}$ $THρ_{suppr}$ $THρ_{disc}$ $THρ_{acct}$
and initialized FDB(d,p)

Output: publishable table T*

1. if (n< $THρ_n$) then return with 1
2. for every ti.s where ti.sw=1 & ti.sdf=0 a new QIDB is created containing ti.s and ti.q.
3. ini_itr=0, accept_flag=0 and gen=first G(T)
4. while (ini_itr< $THρ_{iter}$ and accept_flag=0)
5. QIDB blocks are removed if CF() returns true then check the number of QIDB if it is equal to zero then accept_flag=1
6. itr=itr+1 and gen=next G(T)
7. if accept_flag=0 then invoke supp() & dis()
8. check number of QIDB if it is equal to zero accept_flag=1
9. publish T* if accept_flag=1

The resultant anonymization after applying Personal Anonymization of one of the QIDB with $THρ_{acct}=0.1$

System Design

Modules

- Data fetch module
- FDB generation
- QIDB generalization algorithm
- Sdf-Sw Anomyzation Algorithm

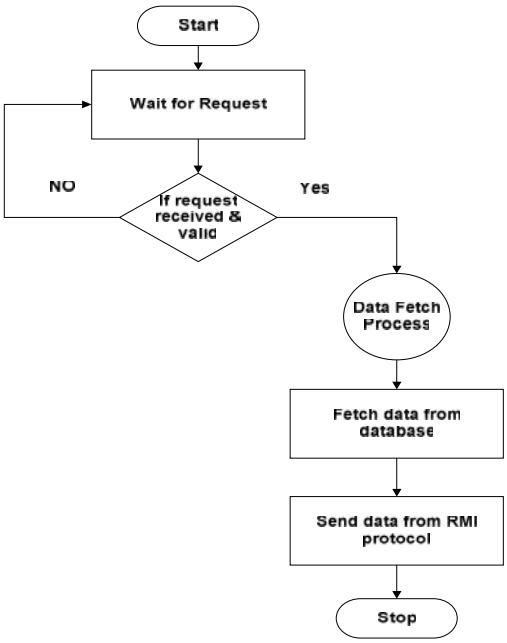

Figure 3 Data Flow of data fetch Algorithm

Implementation

Sw-Sdf Pseudo Code

Input

Private data T with SW-SDF, threshold values n, iter, suppr, disc, THacct and initialized FDB(d,p).

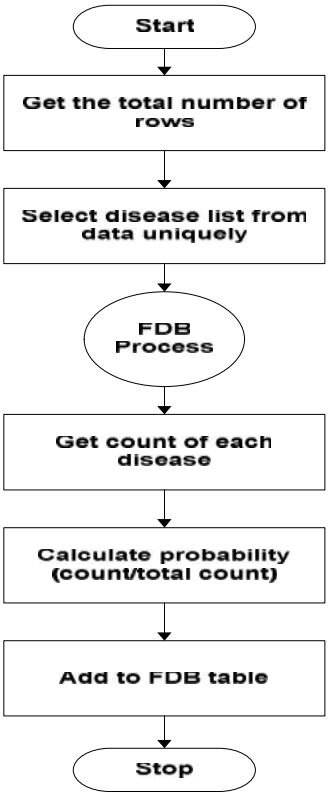

Figure 4 Data Flow Diagram Of FDB Algorithm

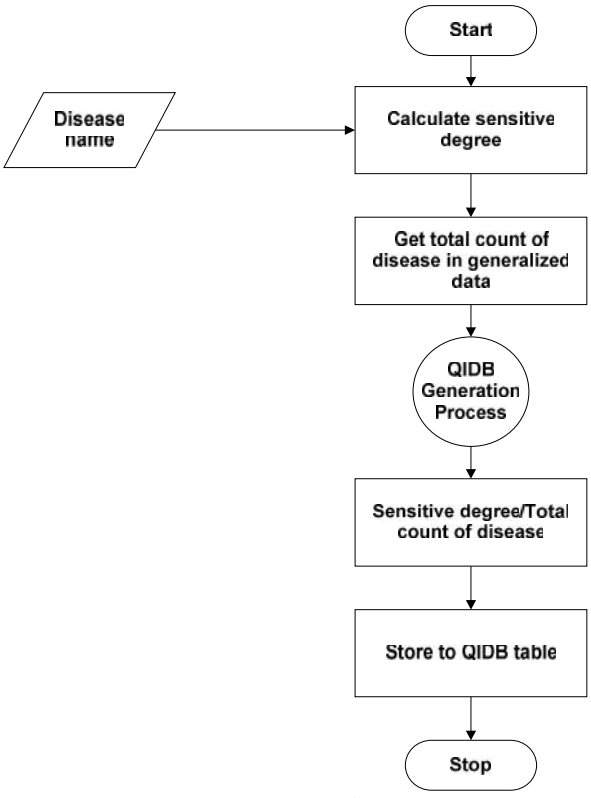

Figure 5 Data Flow Diagram Of QIDB Algorithm.

Output

Publishable table *T**

1. if (*n< n*) then return with *1*
2. for every *ti.s* where *ti.sw=1* & *ti.sdf=0* a new *QIDB* is created containing *ti.s and ti.q* .
3. ini_itr=0, accept_flag=0 and gen=first G(T)
4. while (ini_itr< *iter*and accept_flag=0)

5. QIDB blocks are removed if CF() returns *true* then check the number of QIDB if it is equal to zero then accept_flag=1
6. itr=itr+1 and gen=next G(T)
7. if accept_flag=0 then invoke supp() & dis()
8. check number of QIDB if it is equal to zero accept_flag=1
9. publish T* if accept_flag=1

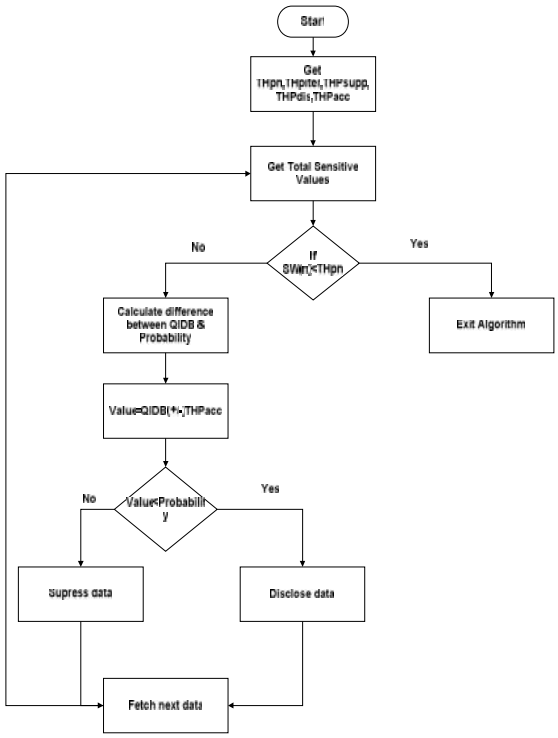

Figure 6 Data Flow Diagram Of Annomysation Algorithm

Mapper Pseudocode

```
def map(key,value):
list=[ ]
for x in value:
if test:
List.append((key,x))
Return list
```

Reducer Pseudocode

```
def reduce(key, listOfValues) :
result=0
for x in listOfValues :
result +=x
return (key, result)
```

RESULTS

Data Provider

Figure 7 Upload Patient Data By Manual Entry

Figure 8 Upload Patient Data By Uploading Database File

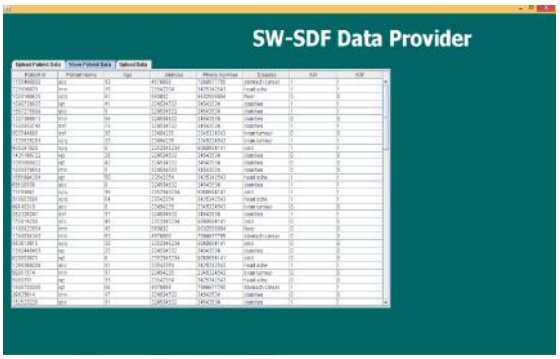

Figure 9 Show Database From Data Provider

Data Anamyzer

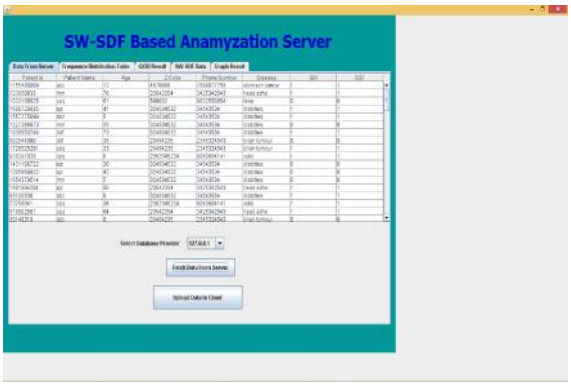

Figure 10 Fetching Data From Data Server

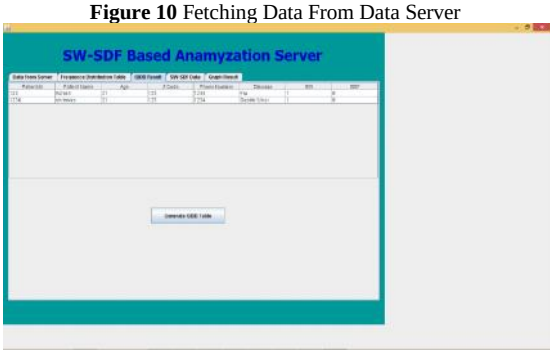

Figure 12 Calculate QIDB And Display Results

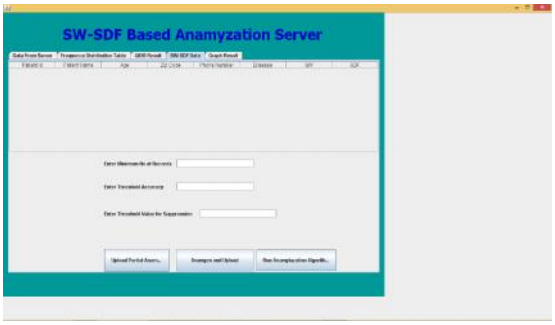

Figure 13 Run Anamyzer And upload To Cloud.

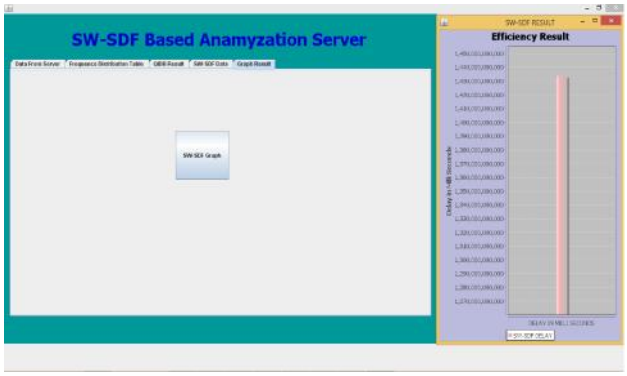

Figure 14 Display Graph Results

Data Retriever

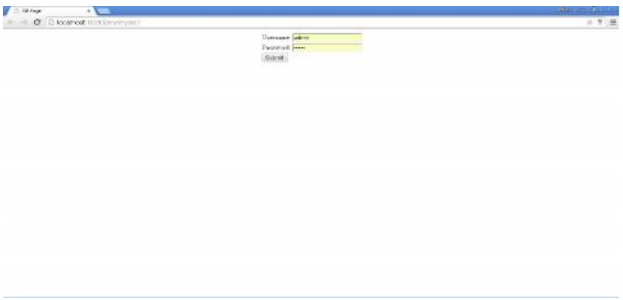

Figure 15 Admin Login Page

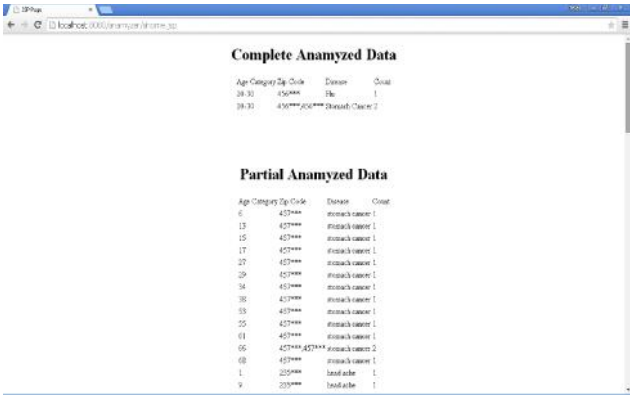

Figure 16 Complete And Partially Anamyzed Data.

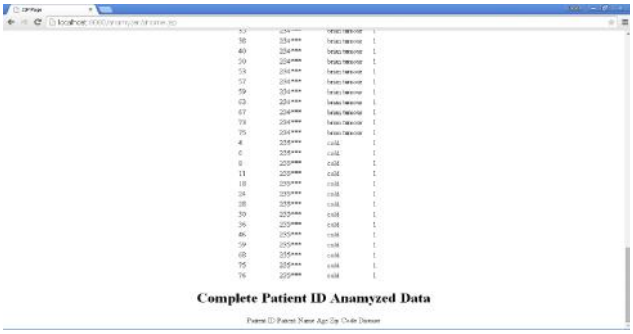

Figure 17 Complete Patient ID Anamyzed Data

CONCLUSIONS AND FUTURE ENHANCEMENT

Our paper medical application of privacy preservation by big data analysis using Hadoop map reduce framework is, SW-SDF based privacy preserving data classification ensures the privacy of the individual by using two flags SW and SDF. SW is automatically calculated using statistical inference and according SDF is accepted form user. The classification algorithm indicated works efficiently for creating decision and information loss is also less. In this paper we have used a DBB tree indexing technique for faster retrieval and classification. Future work includes developing SW-SDF for other mining methods. In our example we have considered only one sensitive attribute; methods must be defined for multiple sensitive attributes. Developing new measures for SW-SDF personalized privacy preservation data mining methods can be investigated.

References

1. Fung B. C. M., Wang K. and Yu P. S, 2007. Anonymizing classification data for privacy preservation, IEEE Trans. Knowl. Data Engin, pp. 711-725.
2. Fung B. C. M., Wang K. and Yu P. S, 2010.Top-down specialization for information and privacy preservation, In Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE). 205-216.
3. Iyengar V. S, 2002. Transforming data to satisfy privacy constraints, In Proceedings of the 8th ACMSIGKDD. ACM, New York, pp. 279-288.
4. L. Sweeney, 2002. Int'l J. Uncertain. Fuzz, k-Anonymity: A Model for Protecting Privacy, vol. 10, no. 5, pp. 557-570.
5. Lefevre K., Dewitt D. J. and Rama krishnan R, Incognito: Efficient full-domain k-anonymity, In Proceedings of ACM SIGMOD. ACM, New York, pp. 49-60, 2005.
6. Machanavajjhala A, Gehrke J, Kifer D and Venkita subramaniam M, 2006. ldiversity: Privacy beyond k-anonymity, In Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE).
7. Ninghui Li , Tiancheng Li , Suresh Venkata subramanian, 2007. t-Closeness: Privacy Beyond k-Anonymity and l–Diversity, ICDE Conference.
8. P. Samarati, Protecting Respondent's Privacy in Microdata Release. IEEE Trans. on Knowledge and Data Engineering (TKDE), vol. 13, no.6, pp. 1010-1027, 2001.
9. Samira Daneshyar and Majid Razmjoo, 2012. Large-scale data process using mapreduce in cloud computing environment, *International Journal on Web Service Computing (IJWSC)*, December.
10. Xiao X. and Tao Y, 2006. Personalized privacy preservation, In Proceedings of the ACM SIGMOD Conference. ACM, New York.