K-ANONYMIZATION IMPLEMENTATION USING APACHE SPARK

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Pratik Tortikar

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

January 2019

Fargo, North Dakota

North Dakota State University
Graduate School

Title

K-ANONYMIZATION IMPLEMENTATION USING APACHE SPARK

By

Pratik Tortikar

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Kenneth Magel

Dr. Bakhtiyor Rasulec

Approved:

March 7, 2019
Date

Kendall Nygard
Department Chair

ABSTRACT

This experiment attempts on data which can reveal a person’s identity to anonymize with k-1 anonymity principle. "Given person-specific field-structured data, produce a release of the data with scientific guarantees that the individuals who are the subjects of the data cannot be re-identified while the data remain practically useful”. The attempt to value the sensitivity and meaningful information with huge amount of data concerning privacy-preserving techniques are maintained to overcome fears with everyone’s delicate data. With this paper, we study the k-anonymity principle algorithm in the context of big data, and introduce a top-down k-anonymization, L-diversity and t-closeness solutions for Apache spark using Java. In the era of volumes of data, science needs more scalable and efficient methods to overcome data leakage, where there is information like public health, diagnosis, sensitive information like name, zip, race, education which leaks the information and would be against privacy of one’s data.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Simone Ludwig, for her continuous support and guidance throughout the project without her feedback and encouraging words I would not have come this far to completion of the project. I’d also like to thank my graduate committee members, Dr. Kenneth Magel and Dr. Bakhtiyor Rasulev for their support with the study. Finally, I wish to acknowledge my family for their unfailing support.

DEDICATION

I would like to dedicate this project to my mother, father, brother and sister who’ve encouraged and supported me through every decision in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

When there is large amount of data involved, be it structured or unstructured or semi-structured, big data tends to solve these problems by different solution approach. It is not just about handling the data, it is about organizing and understanding the complex data which can be analyzed for insights which lead to better decisions and strategic business moves. [1]

While the term “big data” is relatively new, the act of gathering and storing large amounts of information for eventual analysis is ages old. The concept gained momentum in the early 2000s when industry analyst Doug Laney articulated the now-mainstream definition of big data as the three Vs. A high-level overview of big data’s three V’s is presented in Figure 1.

Volume. Organizations collect data from a variety of sources, including business transactions, social media and information from sensor or machine-to-machine data. In the past, storing the data would have been a problem – but new technologies (such as Hadoop) have eased the burden.

Variety tends to different kinds of data present in the market like structured or organized data but with very large volume where normal database cannot handle the operations. And there is text data, unreadable data and other kinds of data which has lots of important information for the business.

Velocity. Data streams at an unprecedented speed must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time.

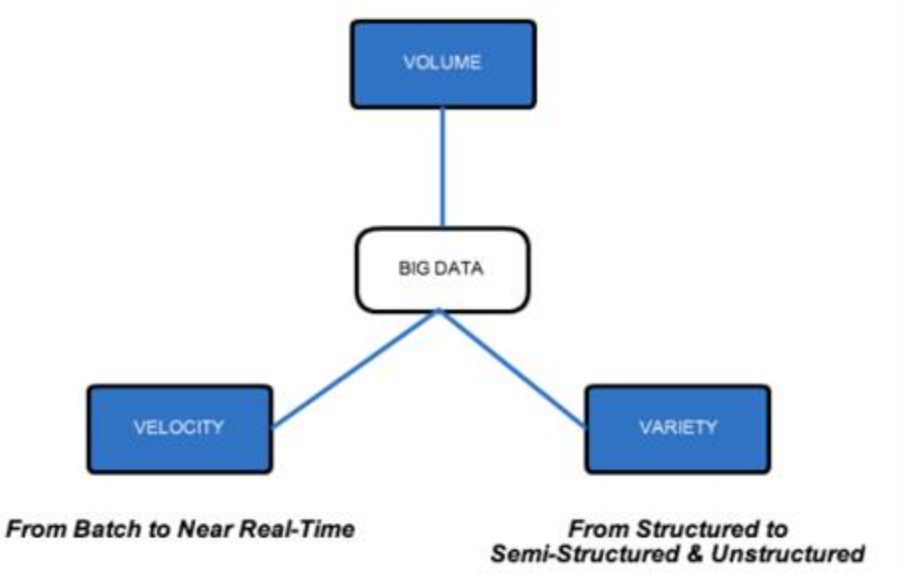


Figure 1. High-level overview of big data with three V's [2]

1.1. Data Management

As computing moved into the Software industry rapidly, data was stored in flat files that imposed no structure. When those organizations needed to get to a level of detailed understanding about customers, they had to apply brute-force methods, including very detailed programming models to create some value. Later things changed with the invention of the relational data model and the relational database management system (RDBMS) that imposed structure and a method for improving performance. Most importantly, the relational model added a level of abstraction (the structured query language (SQL), report generators, and data management tools) so that it was easier for programmers to satisfy the growing business demands to extract value from data. Data management was all about to try and solve a specific type of data related problem. Each of these problems or phases evolved because of cause and effect, for example, a new technology or solutions are introduced in the market, they require the discovery of new approaches/advices. When they are first introduced, the approaches needed a set of tools to allow managers to study the relationship between data elements. When companies started storing semi-structure and unstructured data, analysts needed new capabilities such as natural language–based analysis tools to gain insights that would be useful to business, to gain value from that data required new innovative tools and approaches [3].

The relational model of the database has an ecosystem of tools from a large number of emerging software products. It filled a growing need to help companies better organize their data and be able to compare transactions from one geographical location to another. In addition, it helped business managers who wanted to be able to examine information such as inventory and compare it to customer order information for decision-making purposes. But a problem emerged from this exploding demand for answers: Storing this growing volume of data was expensive and accessing it was slow. Making matters worse, lots of data duplication existed, and the actual business value of that data was hard to measure. [4].

Big data is a traditional term for the strategies and technologies needed to collect, organize, process, and form meaningful insights from large datasets. While the problem of working with data that exceeds the computing power or storage of a single computer is not new, the pervasiveness, scale, and value of this type of computing has greatly expanded in recent years.

In this per say "large dataset" means a dataset too large to reasonably process or store with traditional tools or on a single computer which a relational Database cannot process or handles the volume of real time data analysis; which means that the common scale of big datasets is constantly shifting and may vary significantly from organization to organization. [5]

1.2. Clustered Computing

When considering the qualities and large quantities of big data, individual computers are usually incapable of computing the data at most of the stages due to limited memory management and speed. To better handle these kinds of huge data, computer clusters like distributed systems are beneficial. Several smaller computers are combined or clustered which can provide data computing or management of resource pooling, high availability, fault tolerance, maintainability and scalability.[6]

The general clustering technique in distributed data fashion is shown is Figure 2. It often acts as a foundation which other software interfaces process the data. The computers involved in the clustering are also typically involved with the management of a distributed storage system with different clustering techniques. [7]

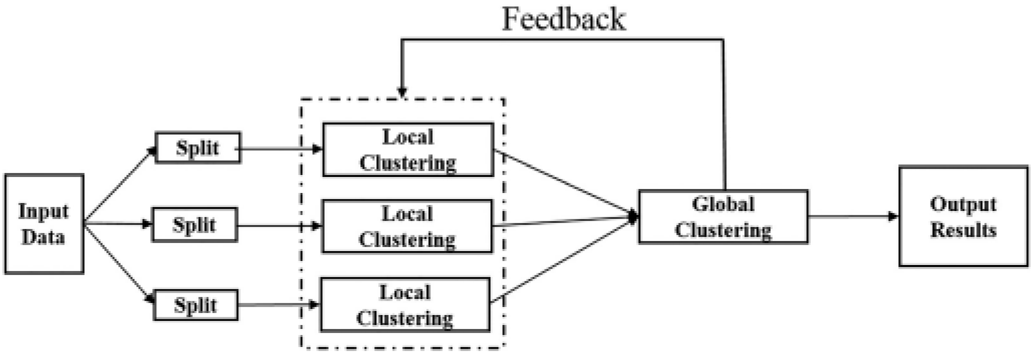


Figure 2. General clustering for big data [8]

2. OVERVIEW ON BIG DATA

Big Data technologies are important to analyze more data, which can lead to better decision-making cost reductions, lesser risk for the business, and efficiency. To enhance the power of big data, we would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real-time and can protect data privacy and security. [9]

2.1. Traditional Approach

The traditional approach is presented in Figure 3. In this approach, a computer might be deployed to store and process big data, where the large volumes of data is stored in relational database management systems like an Oracle database or a Microsoft SQL server, etc. And software to interact with these databases or servers. It has a limitation where these kind of databases works well when there is less volume of data which are usually accommodated by traditional database servers or has limits on the processor in it. When it comes to dealing with huge volumes of data, it is considered a tedious task to analyze and process such volumes of data. [10]

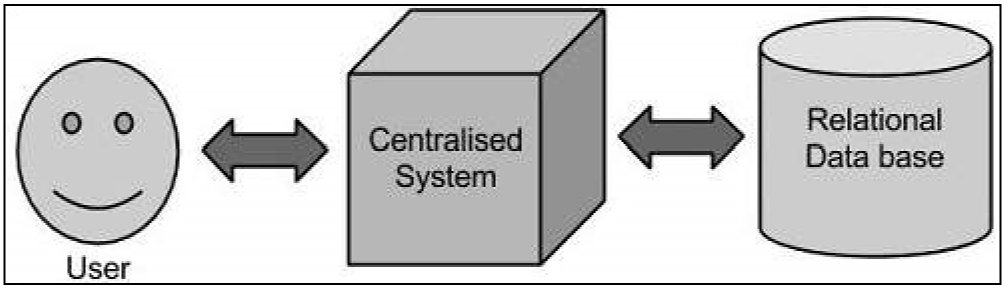


Figure 3. Traditional approach of a big data before hadoop [11]

Google came up with a solution using an algorithm called MapReduce programming, which divides bigger tasks into smaller chunks and assigning those to individual computer in a cluster connected over a network and collects the result to a final dataset.

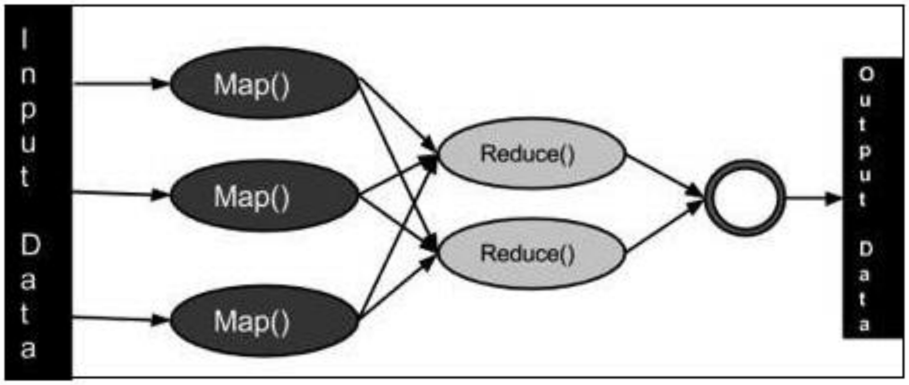


Figure 4. MapReduce basic algorithm flow [12]

The basic MapReduce algorithm flow is presented in Figure 4. With the solution of big data using the MapReduce algorithm an open source project was started called Hadoop by Apache. Hadoop runs applications using MapReduce algorithms, where the data is processed in a distributed cluster across a network. The Hadoop framework is capable to develop, analyze and process applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amount of data.

3. APACHE SPARK

Apache Spark was developed a faster alternative to Hadoop. Fact that Apache Hadoop reads and writes data from the disk which apparently slows down the process, Spark on the other hand stores the data in-memory and reduces the read-write cycles which makes Spark 100 times faster than Hadoop. [13] Spark is known to design and to cover a wide range of workloads which previously required separate distributed systems, including batch applications, iterative algorithms, interactive queries, and streaming which is necessary in data analysis. Spark can also run on Hadoop clusters and access any Hadoop data source, including Cassandra (NoSQL database management system), Yarn, Pig, Hive, etc.

Spark’s core execution system is built in different languages like Scala, Python, Java and R. It also has capabilities to run various machine learning algorithms using Spark MLlib.

3.1. Spark Eco-System

Spark Core component is the foundation of distributed processing of large datasets. It also has responsibilities of all basic spark functions such as input/output, scheduling and monitoring the jobs on clusters, tasks, networking with various storage systems, which has fault tolerance and efficient memory management. [14] This is responsible to deliver speed by providing in-memory computation capability.

3.2. Resilient Distributed Dataset

Spark core is also embedded with a special type of collection named as Resilient Distributed Dataset commonly known as RDD. RDD are among the abstraction of Spark which handles partitioning the input data among the available clusters by dividing them equally or with respect to the size. It basically has two operations which are Transformations and Actions.

Transformation: These are the functions which produces new RDD from an existing RDD.

Actions: These produce non-RDD values only after the transformations are performed, and returns a value of any type but RDD.

3.3. Apache Spark SQL

Apache Spark SQL is a distributed framework for structured data analysis and processing which helps in computation and information on the structure of the data, with use of distributed SQL query engine.

3.4. Spark Streaming

Spark allows fault tolerant stream processing of live data streams or also called as real-time data. It uses micro batching techniques which allows the task to treat a data stream as a continuous sequence of small batches of data. [15]

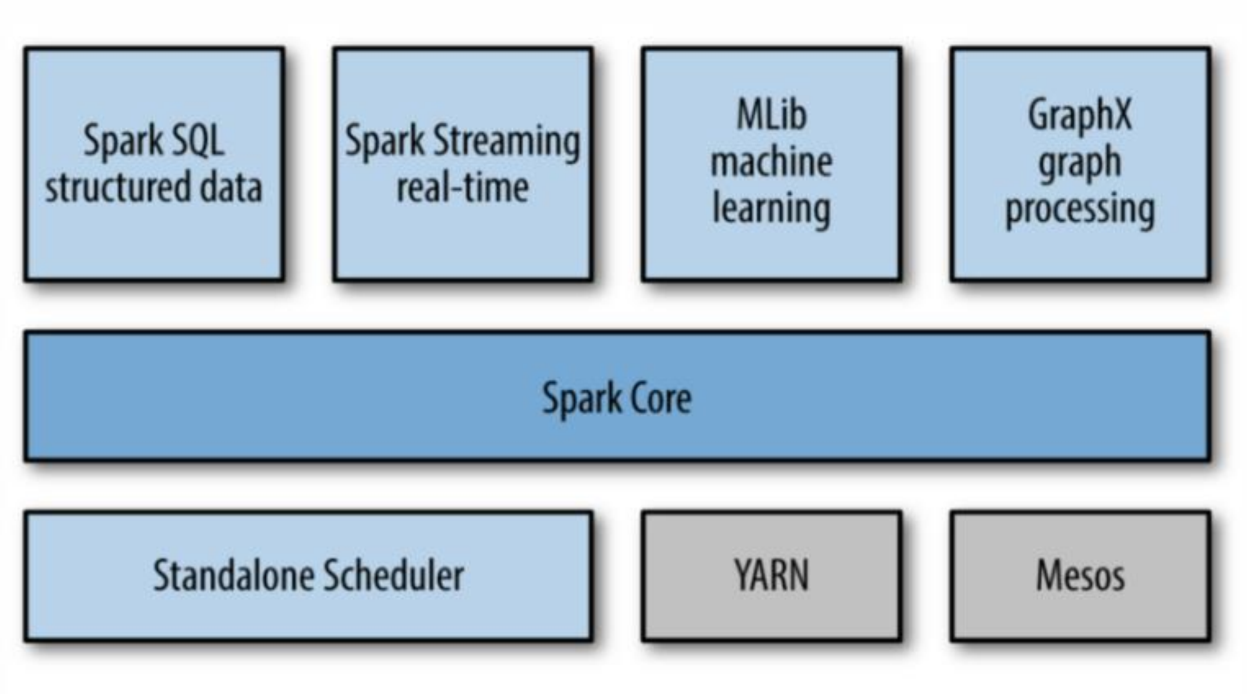


Figure 5. Brief architecture to spark [16]

The architecture of Apache Spark is shown in Figure 5 with spark core and different type of data handling and processing. Apache Spark boosts the existing Bigdata tools for analysis rather than reinventing data analytics. The Apache Spark Ecosystem Components makes it more popular than other Bigdata frameworks. Hence, Apache Spark is a common platform for different types of data processing. For example, real-time data analytics, structured data processing, graph processing, etc. [17]

In this paper, we implement the K-Anonymization algorithm using Apache spark and Java to analyze the data, process it according to the algorithm. The algorithm is implemented in two modes which are local mode cluster, and GPU clusters with partitions, which can be compared to check the efficiency and performance.

4. K-ANONYMITY ALGORITHM

The tremendous growth of data including personal data that can be collected and analyzed for research purposes is retrieved from various repositories. Data mining tools were most often used to analyze these kinds of data which infer trends and patterns. [18] With these speeds of growth in data more and more attention is given to privacy protection of data being released. Therefore, when releasing the data, we should ensure more privacy and integrity as much as possible [19] and the use of data containing personal information must be restricted in order to protect individual privacy. One possible solution is that instead of releasing the entire database of information on people or companies or health care, the database owners can aggregate or filter the data by eliminating or hiding private sensitive information. [20] However, many of the data-mining tasks are not much reliable when privacy of data is a concern and researchers need to extensively analyze the data to discover data aggregation queries of interest. In such cases, query auditing and secure function evaluation techniques does not provide complete solution, as we need to release an anonymized view of the database that ensures the non-sensitive query aggregates, perhaps with some error or uncertainty. [21]

K-anonymity is an important model that prevents joining attacks in privacy protection. The Anonymization algorithm makes use of sanitization techniques to hide the exact values of the data. [22] Even after suppressing the sensitive information, the data can still identify attributes and extract identities with respect to private information. For example:

Table 1. The original data with key attribute ruled out [23]

Age	Race	Gender	Zip Code	Diseases
47	white	Male	21004	Flu
35	Hispanic	Female	21004	Migrane
27	Black	Male	58105	Aids

Table 1 shows a high-level representation of generalization after removing the social security number and name which are the most sensitive data to identify someone. When we join this table to a public database like a voter-list then the columns in the above table can be used to identify individuals.

To ensure the data privacy or protection, we adopt the *k*-anonymity model that was proposed by *Samarati and Sweeney* over the simple anonymization algorithm [24]. Suppose we have a table consisting of *n* tuples each having *m* quasi-identifying attributes (Age, Race, Gender and Zip Code in the above table), and let k > 1 be an integer. The *k*-Anonymity algorithm anticipates the generalization of entries in addition to suppression in the table to ensure for each tuple in the output table there must be at least *k-1* other tuples that are identical along the quasi identifiers. For example,

Table 2. Suppressed data over original data [24]

Age	Race	Gender	Zip Code	Diseases
*	*	*	21004	Flu
35	Hispanic	*	*	Migrane
*	*	Male	58105	Aids

Table 2 shows the K-Anonymized table where K=2 anonymous dataset that ensures the protection of individual privacy such that, even if the data table has all the quasi-identifying attributes of the individuals it will be very difficult to track down any individuals records than n set of k records. It also prevents record linkages with publicly available database and keeps the individual's records or data hidden in a crowd of k-1. The parameter of k should be chosen depending on the application and data. [25]

4.1. Related Definitions for *k-Anonymization* Algorithm

4.1.1. K-Minimum Generalization after Adding Suppression

Suppose $\boldsymbol{T_i}$ **,** $\boldsymbol{T_j}$ are two data tables and $\boldsymbol{T_i > T_j}$,we set MaxUp as the specified inhibition rate [5]. Then $\boldsymbol{T_j}$ ***is*** $\boldsymbol{T_i}$'s K-minimal generalization if and only if the following conditions are true:

$\boldsymbol{T}_{\mathrm{j}}$ Satisfies K anonymity

There does not exist a $\boldsymbol{T_z : T_i < T_z}$,$\boldsymbol{T_z}$ satisfies k-anonymity and $\boldsymbol{DV_z < DV_i}$

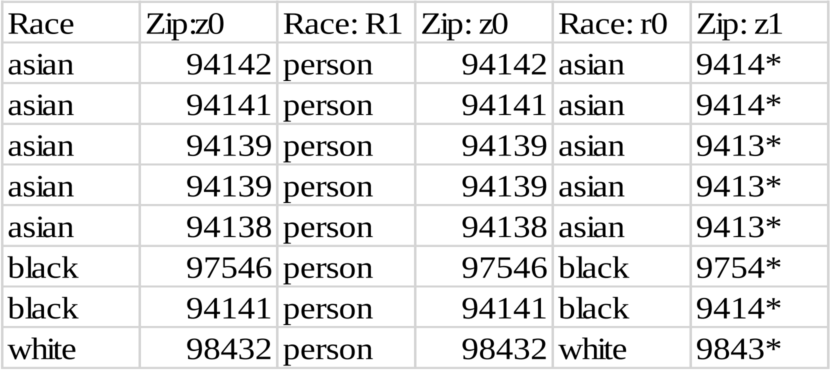
where $\boldsymbol{DV_z}$ and $\boldsymbol{DV_i}$ are the distance vector of $\boldsymbol{T_i}$ ***and*** $\boldsymbol{T_j}$, these conditions state that if generalization $\boldsymbol{T_j}$ is the smallest, then there does not exist a generalization which has a generalization relationship between their distance vectors, or they have the same distance vector and they have a smaller inhibition rate. [26]

4.2. Anonymity Principles

During the dataset publication, failing to preserve the individual privacy information might lead to various consequences. This is the main reason this type of algorithm is being used and the dataset should be anonymized and sanitized. In this section, we include commonly used principles for anonymity related to *k-anonymization.*

K-Anonymity: The attributes in the dataset are noted into four types namely sensitive attributes, Key identifiers, Quasi-identifiers, non-sensitive attributes that does not identify a person necessarily, and other non-sensitive attributes publicly accessible data attributes.

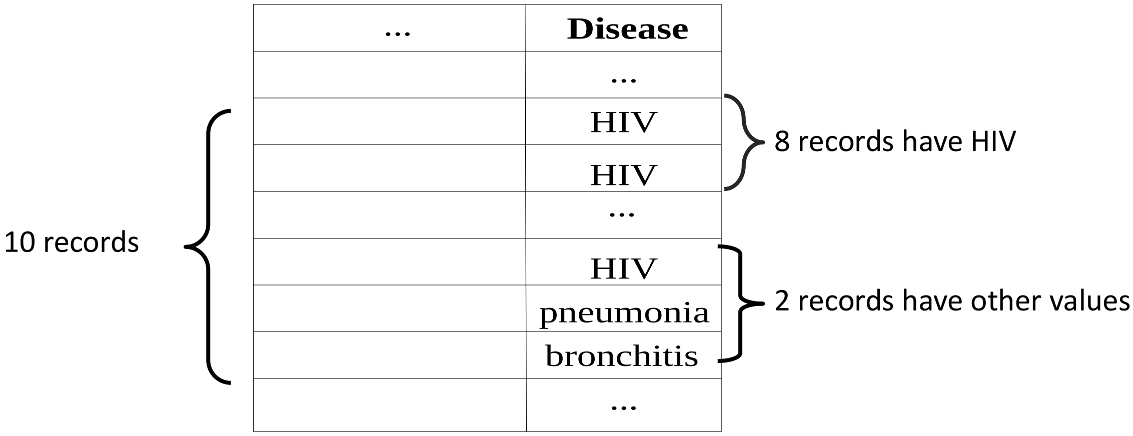
Table 3. Sensitive information vs anonymized information [27]



Race	Zip:z0	Race: R1	Zip: z0	Race: r0	Zip: z1
asian	94142	person	94142	asian	9414*
asian	94141	person	94141	asian	9414*
asian	94139	person	94139	asian	9413*
asian	94139	person	94139	asian	9413*
asian	94138	person	94138	asian	9413*
black	97546	person	97546	black	9754*
black	94141	person	94141	black	9414*
white	98432	person	98432	white	9843*

Table 3 represents sensitive vs anonymized information, in which the zip code can be converted into a common, generalized column, which can be difficult to reveal an individual’s zip code. *l-Diversity*: *k-anonymity* is commonly used to solve only record-linkage kind of privacy attacks and hence lacks to solve all kinds of privacy protection. In a *k-anonymized* dataset the records with the same quasi-identifiers can have the same value for the sensitive attribute. The *L-diversity* privacy principle [28] makes sure that the sensitive attribute has diverse values within every quasi-identifier groups. In other words, *l-diversity* solves the attribute linkage problem. In practice, *l*-diversity is applied as additional security or anonymity to k-anonymization. For example: the race column is being anonymized in Figure 8.

L-Diversity: A table to is said to have *l-diversity* if every equivalence (i.e) considering sensitive attributes) class of the table has l-diverse value (i.e) there must be at least “l” well represented values for sensitive attribute. Figure 7 shows the distinct L-diversity does not prevent probabilistic inference attacks. Machanavajjhala et al. [29] give several interpretations of the term “well-represented”. In Figure 7, the sensitive attributes are in a well-represented form where it is categorized into different values as shown, HIV diseases in one group and other closely related into other groups. These records are chosen with respect to L "well-represented" sensitive attributes. These can be grouped together when Quasi Identifiers are wisely chosen, and these quasi identifiers depend on the dataset being used, as it can be different for different kinds of data.



...	**Disease**
	...
	HIV
	HIV
	...
	HIV
	pneumonia
	bronchitis
	...

Figure 6. Distinct *L-Diversity* with "L" well represented sensitive attribute [30]

Figure 6 shows One sensitive attribute with two values: HIV+(1%)/HIV- (99%). Suppose one class has equal number of HIV+ and HIV- Satisfies any 2-diversity requirement. Anyone in the class has a 50% probability of being HIV+ (compare it to 1% chance in overall population).

Figure 7 represents the generalization of race as an example for t-closeness when the attribute is needed and cannot be ignored.

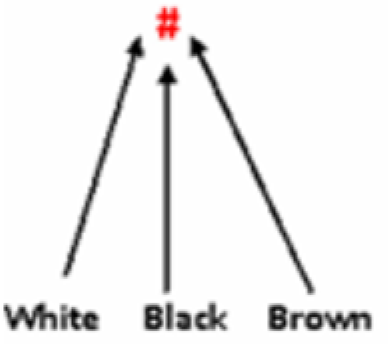
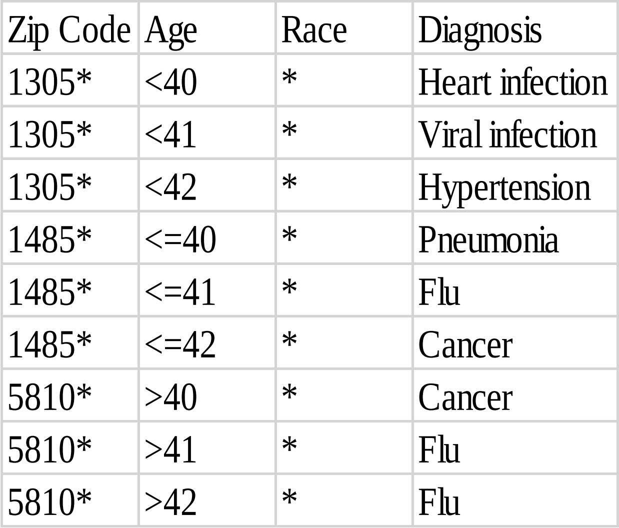


Figure 7. *L-Diversity* example illustration [31]

T-Closeness: *T-Closeness* privacy principle is further refined by the *k-anonymity* and *l-diversity* principles such as rounding up the age column, etc, which is represented in table 4. Suppose there are two aspects "Gender" and "ZIP Code" of a relation *T*. The value of this attribute Gender at level-0 of *T-closeness* can be "Male" and "Female". To consider level 1 of closeness with respect to attribute Gender we must generalize the values, generalizing these two values "Male" and "Female" to another value, say, "Person/ human being". By generalizing the values of attribute Sex to "Person" we achieve level 2 of *T-Closeness*. Similarly is the ZIP code but the level can be increased. By combining different levels of generalization of different attributes we can form the domain. Figure 7 shows an example on how every race is considered into one category without losing the information in the data.

Table 4. Indicates the *L-Diversity* and *T-Closeness* principles [32]



Zip Code	Age	Race	Diagnosis
1305*	<40	*	Heart infection
1305*	<41	*	Viral infection
1305*	<42	*	Hypertension
1485*	<=40	*	Pneumonia
1485*	<=41	*	Flu
1485*	<=42	*	Cancer
5810*	>40	*	Cancer
5810*	>41	*	Flu
5810*	>42	*	Flu

4.3. Materials and Methods

4.3.1. Basic Definitions

a. Sensitive attributes:

Can also be named as Key identifiers depending on the dataset. This tells us the information like name, SSN number, email ID, etc.

b. Quasi-Identifier Attribute:

A quasi-identifier set is a minimal set of attributes in table T that can be joined with external information to de-identify individual records. For example, Table 1 can have all the attributes combining to a quasi-identifier set.

c. Non-Sensitive attributes:

These kinds of data are not included with other attributes does not leak or indicate any sensible information like simple medical records, common occupation.

4.4. Methodology

The basic system architecture of K-Anonymity algorithm is represented in Figure 8. The aim for the experiment is to study the anonymity algorithm with l-diversity, t closeness using java and apache spark integrated and comparing the execution of various datasets differed in sizes and the privacy of datasets. The objectives of this study are referred to by the following methodology using 3 steps:

1) Study of the raw dataset
2) Installation of the software,
3) Java code for *K*-anonymity algorithm and output metrics.

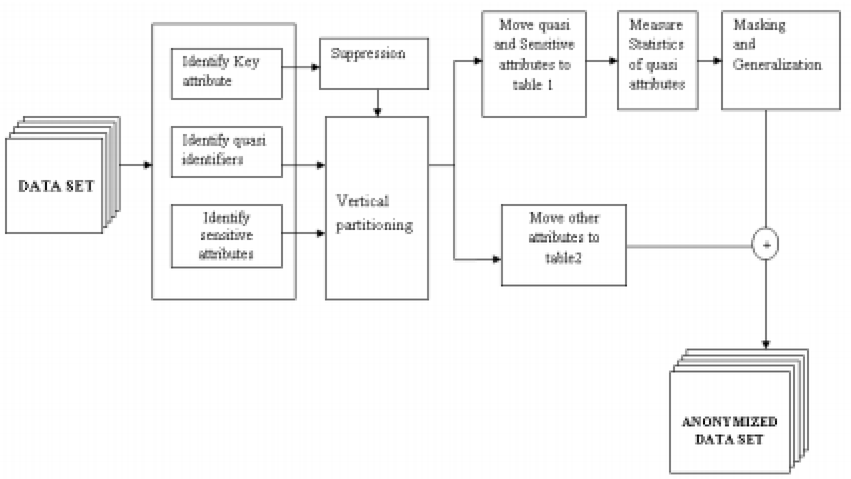

DATA SET
Identify Key attribute
Identify quasi identifiers
Identify sensitive attributes
Suppression
Vertical partitioning
Move quasi and Sensitive attributes to table 1
Measure Statistics of quasi attributes
Masking and Generalization
Move other attributes to table2
+
ANONYMIZED DATA SET

Figure 8. Basic system architecture of generalized *K-Anonymity* algorithm [33]

5. EXPERIMENTS

To experiment with this algorithm, we need to install certain software and other requirements. This section discusses about all the factors involved in this experiment. For local machine experiment, we install Windows operating system [34], along with Java [35]. We have used Microsoft Visual Studio Code to write the Java code [36], which can run on any operating system. Apache Spark is a framework we have used, which is the main reason for this experiment to study on. We do not need to install any third-party software to run java application using Apache spark [37]. The Spark shell is used to run all the application or jobs which is written in Scala or Java. Once all these software's are installed, we need to set the appropriate environment variables to integrate these to be used on single application alongside, for this we use path variables which are commonly referred as variables available in operating system that specifies the directories or path where the executable programs are written and store.

5.1. Datasets Overview

The Adult dataset [38] used in this project has 17,012,800, 8,506,400, 5,103,840, 4,253,200 records for 2gb,1gb, 600mb and 500mb, respectively. It has a binomial label indicating a salary of <50K or >50K USD. In this dataset, 76% of the records in the dataset have a class label of <50K. There are total of 14 attributes which consists of 8 categorical and 6 continuous attributes. The employment class represents the type of employment such as self-employment, private offices or federal and occupation describes the employment type such as farming, field works, clerks or in management. Education mentions the highest level of education attained such as bachelors, masters or doctorate. The relationship attribute has categories such as unmarried or husband and marital status has categories such as married or separated. The other nominal attributes are country of residence, gender and race. People with similar demographic characteristics should have similar weights, etc.

The census income dataset [39] used in this project has 3,990,460, 1,995,230, 1,197,138, 99,761 records for 2gb,1gb, 600mb and 500mb, respectively. This data set contains weighted census data extracted from 1994 and 1995 from the population surveys conducted by the U.S. Census Bureau. It contains 41 different demographic and employment related variables. The instance weight indicates the number of people in the population that each record represents due to stratified sampling.

5.2. Data Cleaning and Expansion

We have expanded the datasets Adult and Census Income by multiplying the records to 500MB, 600MB,1GB and 2GB, respectively.

6. IMPLEMENTATION

For experimenting, we used two different datasets Adult and census income with k=4 anonymity value to compare the K-Anonymized algorithm with L-diversity where L>=2 value and t-closeness where T=2 using Apache spark. The adult dataset is computer from UCI machine learning repository. The Adult dataset contained 28,961 rows and several attributes as columns. And the census income dataset contained 48,888 rows and different set of attributes. The experiments are performed on a local machine first with 8GB memory and 500GB hard disk and on GPU machine with configuration: Nvidia Tesla K40 with 12GB of global memory, 2,880 stream processors memory bandwidth of 288 GB/sec.

6.1. Steps

1) Generate all generalizations of the private table
2) Discard those that violate k-anonymity
3) Find all generalizations with the highest precision
4) Return one based on some preference criteria (worst case)

We have used Java code to implement the anonymity algorithm in Apache spark using Visual studio code. We would like to emphasize the internal parameter of data partitions and worker nodes to compare the different execution time and efficiency with respect to the output parameters generated.

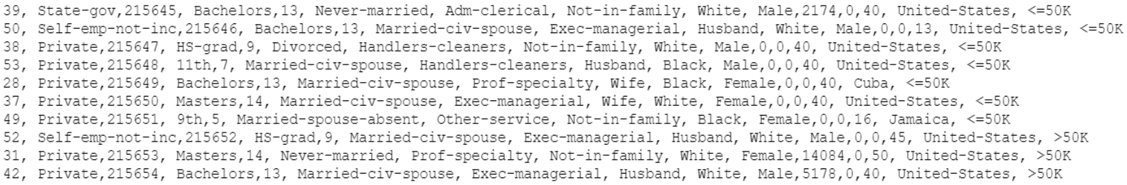
An adult dataset contained different columns specifying information with k=4 anonymity value, L>=2 and T=2 on medical records from health insurance data from a region where Race, Birth Year, Gender and Zip Code together constitute the quasi-identifier and diagnosis is the sensitive attribute, and, we have ruled out the key attributes such as name, email ID and SSN number.

With Census-income dataset pertaining columns which specifies information on income of the individual from certain region with their personal information like Age, Work Class, Zip Code, Education, Education-Num, Marital Status, Occupation, Relationship, Race, Sex, Balance, Unknown, Hours Per Week, Country, Salary where other key attributes where ruled out in data cleaning process like name and work company with quasi identifiers such as age, work class, zip code, education, marital status.

6.2. Results

6.2.1. Adult Dataset Experimental Output

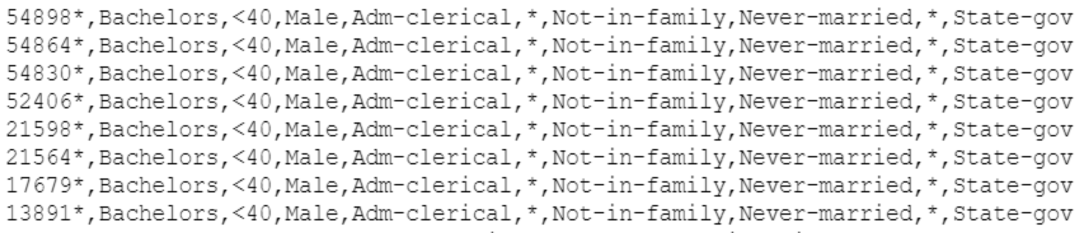
The Adult data set was extracted in 1994 from general data of the United States. It contains continuous and nominal attributes, describing some social information (age, race, sex, marital status) about the citizens registered. The original dataset from UCI library after cleaning and removing the unwanted attributes has several columns out of which the quasi identifiers are age, zip code, race, place, occupation is shown in Figure 9.



```
39, State-gov,215645, Bachelors,13, Never-married, Adm-clerical, Not-in-family, White, Male,2174,0,40, United-States, <=50K
50, Self-emp-not-inc,215646, Bachelors,13, Married-civ-spouse, Exec-managerial, Husband, White, Male,0,0,13, United-States, <=50K
38, Private,215647, HS-grad,9, Divorced, Handlers-cleaners, Not-in-family, White, Male,0,0,40, United-States, <=50K
53, Private,215648, 11th,7, Married-civ-spouse, Handlers-cleaners, Husband, Black, Male,0,0,40, United-States, <=50K
28, Private,215649, Bachelors,13, Married-civ-spouse, Prof-specialty, Wife, Black, Female,0,0,40, Cuba, <=50K
37, Private,215650, Masters,14, Married-civ-spouse, Exec-managerial, Wife, White, Female,0,0,40, United-States, <=50K
49, Private,215651, 9th,5, Married-spouse-absent, Other-service, Not-in-family, Black, Female,0,0,16, Jamaica, <=50K
52, Self-emp-not-inc,215652, HS-grad,9, Married-civ-spouse, Exec-managerial, Husband, White, Male,0,0,45, United-States, >50K
31, Private,215653, Masters,14, Never-married, Prof-specialty, Not-in-family, White, Female,14084,0,50, United-States, >50K
42, Private,215654, Bachelors,13, Married-civ-spouse, Exec-managerial, Husband, White, Male,5178,0,40, United-States, >50K
```

Figure 9. The original “adult” data Set from UCI Library after data cleaning

Figure 10 shows the output data stored in the disk after running the original dataset form k-Anonymizer algorithm to get an anonymized dataset. This output generally stores only the first 100 records when run through apache spark, but we have increased the result set size and store the entire dataset in the output folder. Also, the output dataset contains different age groups like <40, >40, >20 and <40 etc.



```
54898*,Bachelors,<40,Male,Adm-clerical,*,Not-in-family,Never-married,*,State-gov
54864*,Bachelors,<40,Male,Adm-clerical,*,Not-in-family,Never-married,*,State-gov
54830*,Bachelors,<40,Male,Adm-clerical,*,Not-in-family,Never-married,*,State-gov
52406*,Bachelors,<40,Male,Adm-clerical,*,Not-in-family,Never-married,*,State-gov
21598*,Bachelors,<40,Male,Adm-clerical,*,Not-in-family,Never-married,*,State-gov
21564*,Bachelors,<40,Male,Adm-clerical,*,Not-in-family,Never-married,*,State-gov
17679*,Bachelors,<40,Male,Adm-clerical,*,Not-in-family,Never-married,*,State-gov
13891*,Bachelors,<40,Male,Adm-clerical,*,Not-in-family,Never-married,*,State-gov
```

Figure 10. The output generate dataset after running through the algorithm

Figure 11 shows the experiment on the Adult dataset with local cores 4 with driver memory 16g, 22g and 32g. As we can see, the run time for this algorithm on 4 cores has a longer run time, also there are different run times for different driver memory with respect to data partitions. When we use the 2gb dataset with 17,012,800 records the time taken for the GPU machine with 4 nodes using 16g driver memory is real 63m50.299s, user 16m11.968s and sys 2m49.224s. Here, we can see the improvement when we use more partitions and nodes with increasing data sizes.

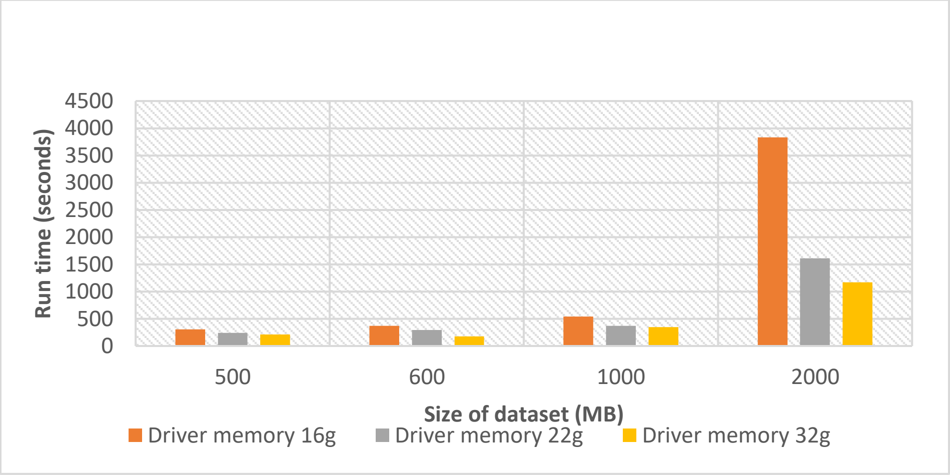


Figure 11. Bar graph of result with 4 cores – adult data set

The above bar graph in Figure 11 represents the execution time taken for the adult data set with minimal driver memory has a greater execution time for Adult Data Set: K=4, L>=2 And T=2

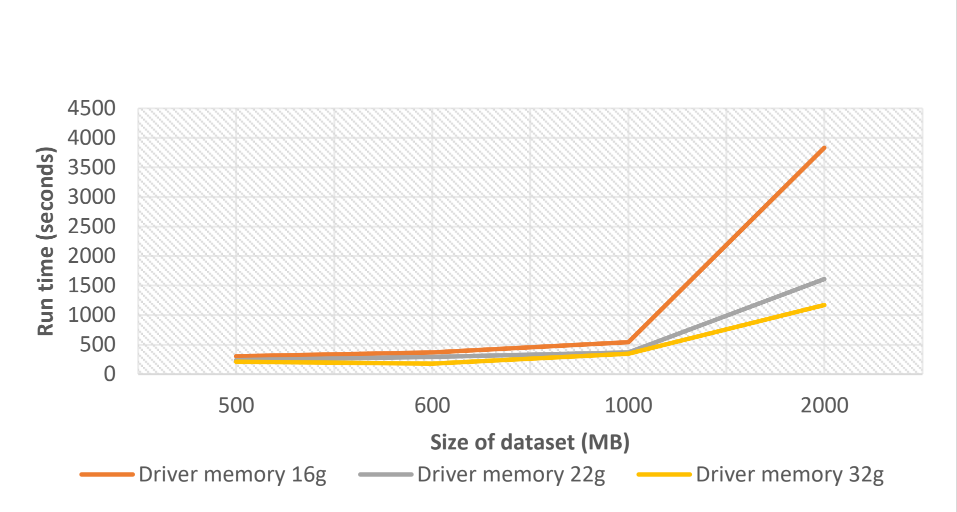


Figure 12. Line graph shows more accurate time span taken for the first adult dataset

Figure 12 shows the line graph of the results from the experiment on the Adult dataset with driver memory 16g, 22g and 32g with a local core number of 8.

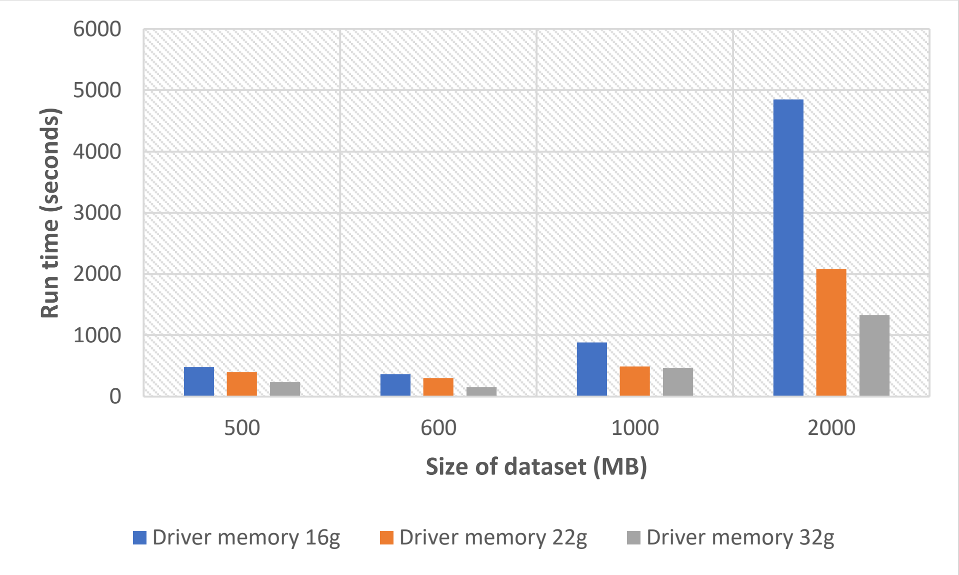


Figure 13. Bar graph of result with 8 cores – adult data

The above bar graph with k=4, L>=2 and T=2 in Figure 13 with 8 cores partitioned has a significant change in the execution time for different scenarios.

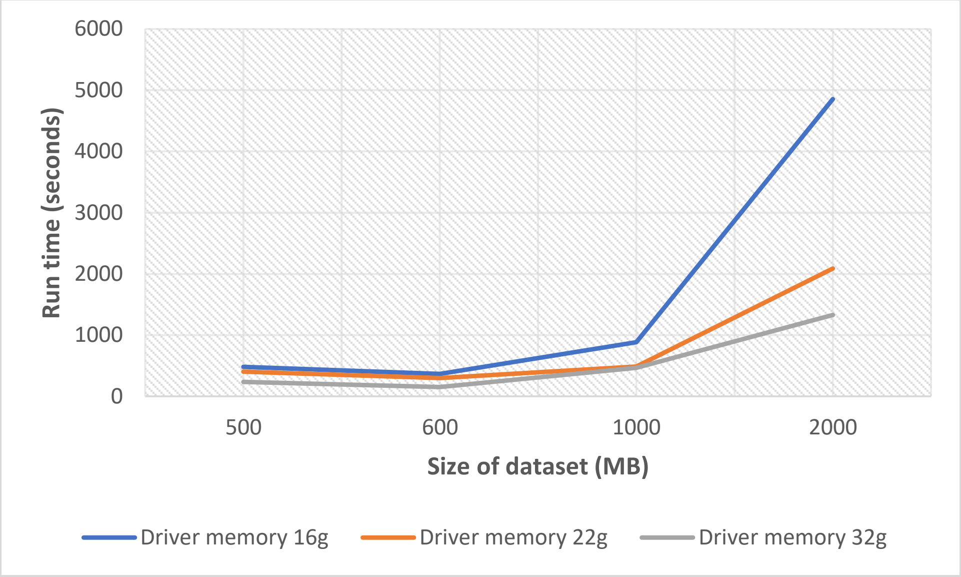


Figure 14. Line graph shows more accurate time span taken for the first adult dataset

The line graph in Figure 14 represent the overall growth in the larger dataset with 16g as the driver memory as compared to minimal change in 22g and 32g driver memory.

6.2.2. Census Income Data Experimental Output

Figure 15 shows the results of the experiments on the census dataset for local cores equals 4 with driver memory 16g, 22g and 32g. Here, the run time for larger dataset is much higher than for the Adult dataset, which is around 72 minutes. The performance increases with an increase in partitions and cluster nodes. If the 2gb dataset uses 16g driver memory with less partitions the run time may increase by 40-45 mins. This algorithm gives different results for different data sets accordingly.

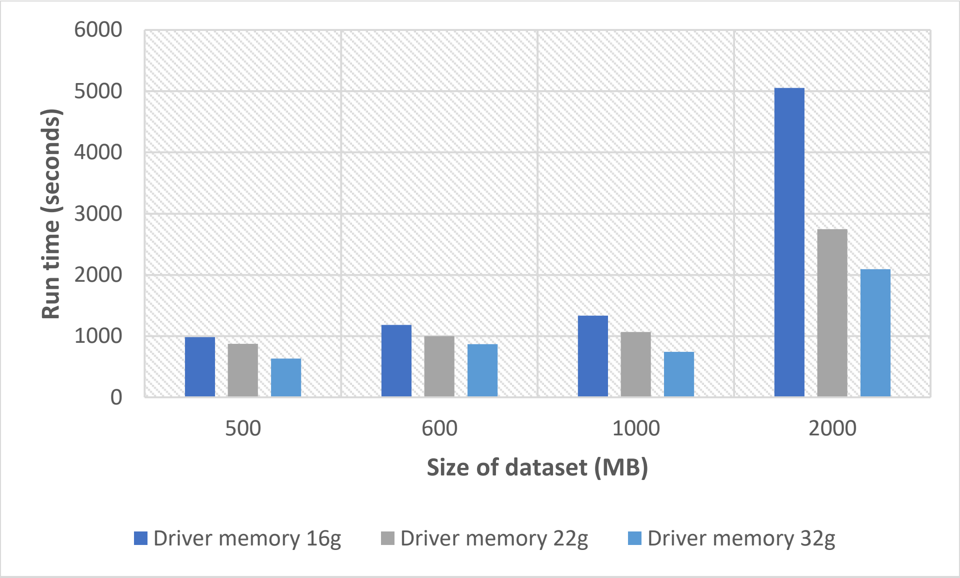


Figure 15. Bar graph of result with 4 cores – census-income data set

The bar graph in Figure 15 explains the consequent increase in execution time with higher amount of dataset with 4 cores.

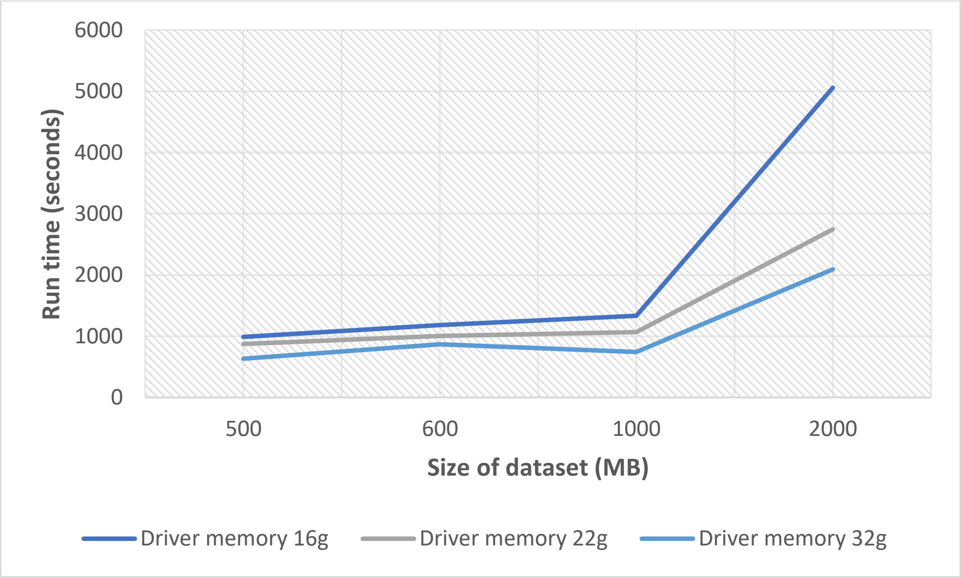


Figure 16. Line graph shows more accurate time span taken for census-income dataset

Figure 16 shows the results of the experiments on the Census dataset with driver memory 16g, 22g and 32g with several local cores of 8. As we can see, the run time for this algorithm with 8 cores is higher. Here depending on the data partitioned within the nodes, the run time varies accordingly. The time taken for 8 nodes and 2gb data with 20 million records is higher which is real 72m04.243s, user 14m10.960s and sys 2m49.224s as compared to 1gb data with 10 million records real12m12.385s, user3m50.036s and sys0m38.144s. Figures 15 and 16 show similar trends.

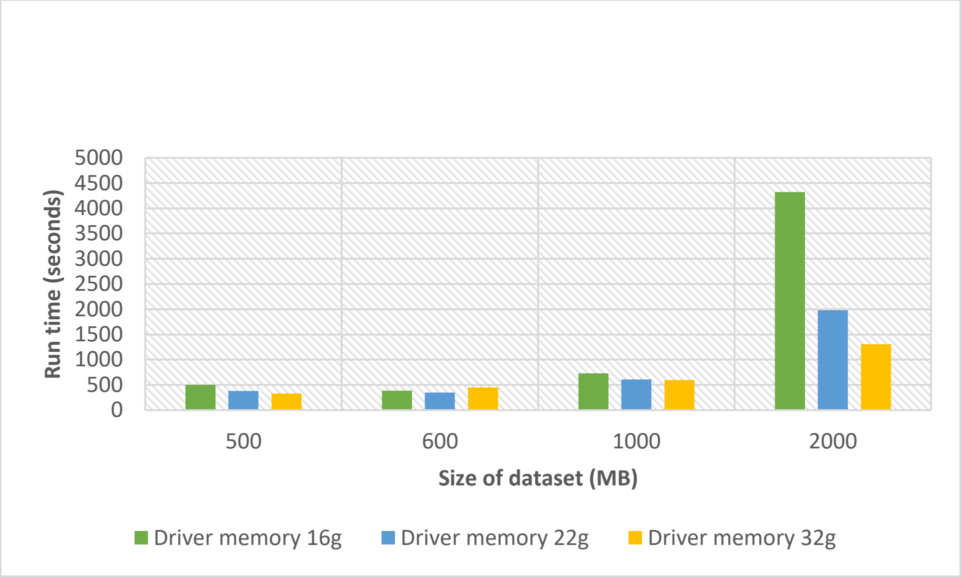


Figure 17. Bar graph of result with 8 cores – census-income data set: K=4, L>=2 and T=2

The graph above in Figure 17 clarifies the different between smaller and larger dataset with same 8 cores.

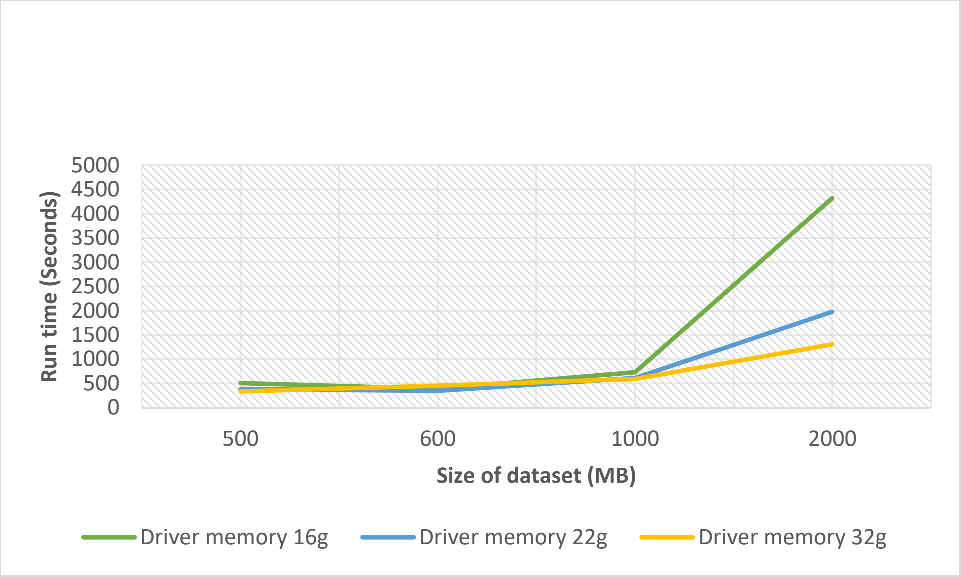


Figure 18. Line graph shows more accurate time span taken for census-income dataset

The line graph in Figure 18 explains that the larger dataset has a much slower execution time with partitions considered with driver memory.

7. CONCLUSION

This is the big data era and the volume and variety of data across the industries are increasing by tera bytes every day. To match this rapid growth of big data processing Apache Hadoop and Spark have been used to analyze and process data in order to achieve high scalability. Because of this approach many algorithms are being implemented using these big data tools to ensure the quality, scalability and security. And of course, privacy and security algorithms are not any different from other applications. Researchers have already implemented various anonymization algorithms on different platforms and in Hadoop. This study used Apache spark and showed how the result can be obtained much faster. We have considered two datasets from the UCI repository with different data and ran the K-Anonymity algorithm, L-Diversity and T-closeness with K=4, L>=2, and T=2 values on both datasets to privatize the data as much as possible with the highest efficiency. At first, the data was cleaned and unnecessary columns were cleaned, and the dataset size was increased from 28,000 records to a few million records with different sizes. We have ran the algorithm with l-diversity and t-closeness concepts along with k-anonymity to ensure the data is anonymous enough to avoid data leak with regards to the columns race, age etc. generalized by using the generalization concept. The efficiency of the algorithm using Apache spark has shown improved results with different size of datasets.

8. REFERENCES

[1] SAS Institute, “What is Big data?”, 2016, Https://Www.Sas.Com/En Us/Insights/Big-Data/What-Is-Big-Data.Html Last Retrieved On October 2018

[2] Xiaomeng Su, “Introduction to Big Data”, 2012, Https://Www.Ntnu.No/Iie/Fag/Big/Lessons/Lesson2.Pdf Last Retrieved on October 2018

[3] Yinghui, Alan Nugent, “Big Data For Dummies,” 2007 Https://Eecs.Wsu.Edu/~Yinghui/Mat/Courses/Fall%202015/Resources/Big%20data%20for%20dummies.Pdf Last Retrieved On October 2018

[4] Judith Hurwitz, Marcia Kaufman, “Big Data,” 2015, Https://Eecs.Wsu.Edul%202015/Resources/Big%20data%20for%20dummies.Pdf Last retrieved on October 2018

[5] Justin Ellingwood, “Big Data Concepts And Terminology”, 1997, Https://Www.Digitalocean.Com/Community/Tutorials/An-Introduction-To-Big-Data-Concepts-And-Terminology” Last Retrieved On October 2018

[6] Neal Barcelo, Nick Legg, “Performance Of Big Data Machine Clusters”, 2000, Https://Personal.Denison.Edu/~Bressoud/Barceloleggbressoudmcurcsm2.Pdf Last Retrieved On October 2018

[7] B. Babcock, M. Datar, “Computer Clustering”, 1997, Http://Infolab.Stanford.Edu/~Ullman/Mmds/Ch7.Pdf Last Retrieved On October 2018

[8] Min Chen, Simone A. Ludwig, “General Clustering for Big Data”, 2017, Https://Pdfs.Semanticscholar.Org/2ab0/D4ded091959f0ed7140b85c90bef49d9ab1b.Pdf Last Retrieved On October 2018

[9] Tutorial point, “Big Data Overview”, 2004, Https://Www.Tutorialspoint.Com/Hadoop/Hadoop_Big_Data_Overview.Htm Last Retrieved On October 2018

[10] Margaret Rouse, “Traditional Approach For RDBMS”, 1996, Https://Searchdatamanagement.Techtarget.Com/Definition/RDBMS-Relational-Database-Management-System Last Retrieved On November 2018

[11] RDBMS, “Traditional Database Management”, 2009, Http://Www.Assignmentpoint.Com/Business/Management/Relational-Database-Design.Html Last Retrieved On November 2018

[12] Tutorial point , “MapReduce”, 2005, Https://Www.Tutorialspoint.Com/Hadoop/Hadoop_Mapreduce.Htm Last Retrieved On November 2018

[13] Suvro Banerjee, “Introduction To Spark”, 2018, Https://Www.Kdnuggets.Com/2018/07/Introduction-Apache-Spark.Html Last Retrieved On November 2018

[14] Dezyre, “Apache Spark Ecosystem”, 2017, Https://Www.Dezyre.Com/Article/Apache-Spark-Ecosystem-And-Spark-Components/219 Last Retrieved On November 2018

[15] Dataflair Team, “Spark Components”, 2016, Https://Data-Flair.Training/Blogs/Apache-Spark Last Retrieved On November 2018

[16] Data Bricks, “What Is Apache Spark?,” 2016, Https://Databricks.Com/Spark/About Last Retrieved On November 2018

[17] Dataflair Team , “Spark Ecosystems”, 2016, Https://Data-Flair.Training/Blogs/Apache-Spark-Ecosystem-Components/ Last Retrieved On November 2018

[18] Gagan Aggarwal, Tomas Feder, “K Anonymity”, 2000, Http://Theory.Stanford.Edu/~Kngk/Papers/K-Anonymity-Jopt.Pdf Last Retrieved On November 2018

[19] Vaidya, J., Clifton, C.: Privacy-Preserving Data Mining: Why, How, And When. IEEE Security & Privacy 2(6), 19–27 (2004) Last Retrieved On November 2018

[20] Kleinberg, Papadimitriou, And Raghavan, “Query Auditing”, 2003, Https://Www.Researchgate.Net/Publication/246388337_How_Auditors_May_Inadvertently_Compromise_Your_Privacy Last Retrieved On November 2018

[21] Gagan, Agarwal, “K Anonymization Principle”, 2000, Http://Stanford.Edu/~Kngk/Papers/K-Anonymity-Jopt.Pdf Last Retrieved On November 2018

[22] Agrawal And Srikant, “Approximation Algorithms For K-Anonymity”, 2000, Http://Theory.Stanford.Edu/~Kngk/Papers/K-Anonymity-Jopt.Pdf Llast Retrieved On November 2018

[23] Krishnaram Kenthapadi, Rajeev Motwani, “Approximation K-Anonymity Algorithm”, 2002, Http://Theory.Stanford.Edu/~Kngk/Papers/K-Anonymity-Jopt.Pdf Last Retrieved On November 2018

[24] Pierangela Samarati, Latanya Sweeney “Protecting Privacy When Disclosing Information: K-Anonymity And Its Enforcement Through Generalization And Suppression”, 2003, Https://Epic.Org/Privacy/Reidentification/Samarati_Sweeney_Paper.Pdf Last Retrieved On November 2018

[25] Samarati And Sweeney, "K-Anonymity", 1996, Https://Dataprivacylab.Org/Dataprivacy/Projects/Kanonymity/Kanonymity2.Pdf Last Retrieved On November 2018

[26] U. Sopaoglu And O. Abul, "A Top-Down K-Anonymization Implementation For Apache Spark," 2017, Https://Ieeexplore.Ieee.Org/Abstract/Document/8258492 Last Retrieved On November 2018

[27] Zhaofeifei, Donglifeng, "K-Anonymity", 2002, Https://Ac.Els-Cdn.Com/S187538921201406X/1-S2.0-S187538921201406Xmain.Pdf?_Tid=E53c2ef3-7fb1-4929-990a14700c3c3c4d&Acdnat=1543684700_4eac80f0041d81f19bea38d5297f701c Last Retrieved On November 2018

[28] N. Li, T. Li, And S. Venkatasubramanian, "T-Closeness: Privacy Beyond Kanonymity And L-Diversity," 2007, Https://Www.Cs.Purdue.Edu/Homes/Ninghui/Papers/T_Closeness_Icde07.Pdf Last Retrieved On November 2018

[29] Jianmin Han, Huiqun Yu, "An Improved L-Diversity Model For Numerical Sensitive Attributes", 2001, Https://Ieeexplore-Ieee-Org.Ezproxy.Lib.Ndsu.Nodak.Edu/Stamp/Stamp.Jsp?Tp=&Arnumber=4685178_Last Retrieved On November 2018

[30] James Joshi, "Information Security & Privacy", 1999, Http://Www.Sis.Pitt.Edu/Jjoshi/Courses/IS2150/Spring15/Lecture12.Pdf_Last Retrieved On November 2018

[31] K. Lefevre, D. J. Dewitt, And R. Ramakrishnan, “Mondrian Multidimensional K-Anonymity,” 2006, https://www.utdallas.edu/~muratk/courses/privacy08f_files/MultiDim.pdf Last Retrieved On November 2018

[32] Ugur Sopaoglu, Osman Abul, “A Top-Down K Anonymization”, 2017, Https://Ieeexplore-Ieee-Org.Ezproxy.Lib.Ndsu.Nodak.Edu/Stamp/Stamp.Jsp?Tp=&Arnumber=8258492 Last Retrieved On November 2018

[33] P. Usha, Shriram Raghunathan, “Anonymity Model For Privacy Preserving Data Mining”, 2004, Https://Www.Researchgate.Net/Publication/258789893_Modified_Anonymity_Model_For_Privacy_Preserving_Data_Mining Last Retrieved On November 2018

[34] Microsoft, “Windows Installation”, 2018, “Https://Www.Microsoft.Com/En-Us/Software-Download/Windows10startfresh” Last Retrieved On February 2018

[35] Oracle, “Java Installation”, 2018, Https://Www.Java.Com/En/Download/Help/Download_Options.Xml Last Retrieved On February 2018

[36] Microsoft, “Visual Studio Documentation”, 2018, Https://Code.Visualstudio.Com/Docs Last Retrieved On January 2018

[37] Apache, “Get Apache Spark”, 2018, Https://Spark.Apache.Org/Docs/Latest/ Last Retrieved On February 2018

[38] Adult Dataset, “Uci Machine Learning Repository.”, 1997, Ftp://Ftp.Ics.Uci.Edu/Pub/ Last Retrieved On February 2018

[39] Census Income Dataset, ”Uci Machine Learning Repository” ,1997, Https://Archive.Ics.Uci.Edu/Ml/Datasets/Census+Income Last Retrieved On June 2018