International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12– No.31, July 2020 – www.ijais.org

A Multi-Nodal Implementation of Apriori Algorithm for Big Data Analytics using MapReduce Framework

Terungwa Simon Yange
Department of Mathematics, Statistics and Computer Science, University of Agriculture, Makurdi, Nigeria

Ishaya Peni Gambo
Dept of Computer Science & Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria

Rhoda Ikono
Dept of Computer Science & Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria

Hettie A. Soriyan
Dept of Computer Science & Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria

ABSTRACT
This paper developed a distributed algorithm for Big Data Analytics to address the delay in the processing of big data. In order to achieve the aim of this research, an inspection of organizational documents, direct observation and collection of existing data from the National Health Insurance Scheme (NHIS) in Nigeria. The algorithm was formulated using Apriori Association Rule Mining and was specified using the enterprise application diagram. The implementation of the prototype for the algorithm was using MongoDB as the big data storage mechanism for the input. Comma Separated Values (CSV) files was used as the storage facility for the intermediate results generated during processing, and MySQL was used as the storage mechanism for the final output. Finally, Apache MapReduce as the big data multi-nodal processing platform and Java programming language as the implementation technology. This prototype was able to analyze different formats of data (*i.e.,* pdf, excel, csv and images) with high volume and velocity. The result showed that the response time was 0.25 seconds, and the throughput was 8865.29 records per second. The stability of the prototype was also evaluated using the confidence of the rules generated. In conclusion, this research has shown that unnecessary delays in the processing of big data were due to the lack of appropriate data analytics tool to enhance the process. This study eliminated these irregularities which paved the way for quicker disbursement of funds to providers and other stakeholders, as well as, a quicker response to requests on enrollment, update and referral.

Keywords
MapReduce, Node, Big Data, Analytics, MongoDB, Apriori

1. INTRODUCTION
The emergence of supercomputers was a significant breakthrough in data processing. However, their high cost hampered their adoption by data engineers. In a bid to effectively handle the increasing demand of users' data processing need at a cheaper cost, computer scientists came up with the distributed computing model. The distributed model served as an alternative to the highly expensive supercomputers [1][2]. As opposed to supercomputers, distributed computing involves a network of large number of computers or entities known as nodes, connected. This architecture provides a high processing capability linking the different number of nodes via a fast network and resource sharing among multiple users at a lower cost as compared to supercomputers. Having multiple nodes processing the same item of data implies that the failure of a particular node in the network will not affect the entire process. With the adoption of web technologies, mobile devices and the reduction of cost of computing infrastructures from the late '90s to date, has led to the explosion of data. To effectively exhume insights form this data for timely and accurate decision making, the performance of analytics against its features is required. Hence, the motivation for the application of distributed computing in the area of data analytics [2].

Data has taken centre stage in every economy in recent time. It is a set of quantitative or qualitative facts about people, things, ideas and events derived from either via measurement or features of items during experiments [3][4]. Data is represented by symbols such as letters of the alphabets, numerals or other special symbols which are suitable for communication, interpretation, or processing by humans or by automatic means. It could either be big or small [3][5][6][7]. Small data refers to a dataset that contains particular attributes (*i.e.,* a dataset with a well-defined structure). It is structured, focused and easily interpreted. In a similar vein, big data refers to very complex data characterized by its high volume, variety, velocity and veracity, as shown in Fig 1. As Fig 1 reflects, it becomes challenging for traditional data management tools, such as MySQL, Oracle, Excel to handle big data [8]. In other words, it is the dataset whose size is beyond the ability of orthodox data processing tools to capture, store, pre-process, and analyze within an acceptable time frame. Big data comes in various categories: structured, semi-structured and unstructured. Both big data and small data are valuable and could turn around an organization if they are effectively handled. Also, the small data is a subset of the big data. Hence, in this research, our focus is on big data which encapsulate all the types of data. The analysis of big data is known as big data analytics. It is the process of collecting, organizing and analyzing large, diverse dataset such as structured/unstructured and streaming/batch with the view to getting meaningful insights that aid in decision making. This was hitherto done with the human brain, which again has lots of limitations. Therefore, considering the volume, velocity and variety of data we are exposed to daily in recent times, it has become challenging to manually do this with the human brain [1].

Furthermore, the need to have an aid that supports the human brain for accurate and effective decision-making process has become more and more inevitable. This support is known as analytics [1][7]. Big data analytics derive meaningful insights from big data which could lead to more self-confidence in decision-making, and better conclusions could be achieved with

higher operational efficiency, and reduction in both cost and risk. It develops strategies that could apply analytics in every facet of human endeavour to answer the questions right in time. Thus, it has been mainstreamed in the human decision-making process. Summarily, big data analytics is premised on making insights available to users, within actionable time, without bothering about the sources and devices used in storing and processing the same. This is where the principles of distributed computing have found a niche in big data analytics. The application of distributed computing in big data analytics can scale the processing and storage of data with an increase in volume, and the ability to use low-cost hardware. This would demystify big data analytics by lowering the cost of analytics, thus, making it affordable by all.

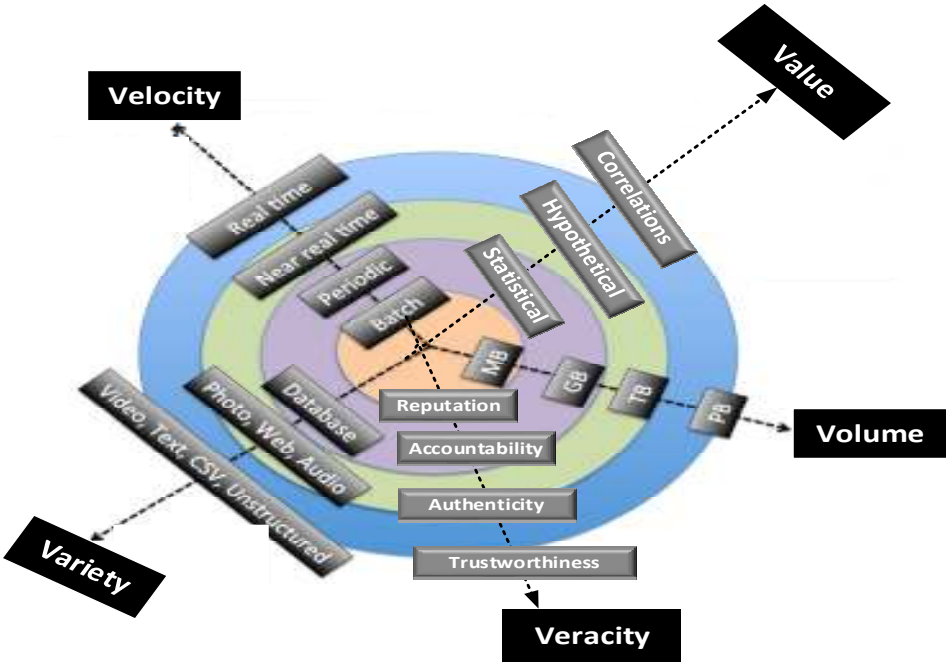

Fig 1: Modified Basic Characteristics of Big Data [8]

To properly reap the dividends of big data analytics, it is wise to apply distributed processing by harnessing the merits of distributed computing. This would involve storage, access, transfer, analysis, visualization using multiple low-cost infrastructures that derive insights within stipulated cost and time to be applied by humans or machines [1]. However, existing works in this area do not cover this aspect of distributed processing in big data analytics. Therefore, this paper aims to fill up this gap. In this research, a distributed algorithm was formulated for the processing of big data using the Apriori algorithm of the Association Rule Mining (ARM). A prototype implementation for the algorithm was done using the MapReduce Framework (provided the multi-nodal platform) and MongoDB (data storage for both structured, semi-structured and unstructured data) as big data technologies, and MySQL and Java Programming Language. The National Health Insurance Scheme (NHIS) data was collected through observation and review of the document to evaluate this prototype using throughput and response time as the performance metrics [2][7].

This paper is organized in the following order. Section one presented the background to the research and also reviewed works that are related and relevant to this study. Section three described methods that are deployed in achieving the purpose of the paper; and the experiments and results are presented in Section three. In Section Four, the interpretation, relevance and limitations of the results presented in Section four are discussed. Section five gives the unique contributions of the paper, limitations of the research and some future research directions as the conclusion for the paper.

2. RELATED WORKS

Distributed computing refers to the use of distributed systems to solve computational problems. Here, a problem is divided into many tasks, each of which is solved by one or more computers [9]. A distributed computing system consists of several processing elements interconnected by a computer network and co-operating in performing specific assigned tasks. Big data technologies include distributed computational systems, distributed file systems, massively parallel-processing (MPP) systems, cloud-based storage and computing, and data mining based on grid computing *etc.* Apache Hadoop is *de facto* software platform that supports data-intensive distributed applications. NoSQL (Not only SQL) database is used for large and distributed data management and database design. Clustering big data is also developing to distributed and parallel implementation [10]. When the volume of data in a database becomes large, it is distributed across different sites. These distributed databases require distributed computing to efficiently store, retrieve and update data in a well-coordinated way [11]. The advent of big data has led to the search for new methods for its storage and analysis. In managing big data, technologies have been created that can use the computing power and the storage capacity of a cluster, with an increase in performance proportional to the number of machines present on the same. One of such technologies is Hadoop, a framework for distributed processing of large datasets across clusters of computers. The Hadoop distributed file system (HDFS) and MapReduce are two critical components of Hadoop. The HDFS handles the storage of big data. MapReduce distributes computing jobs to each server in a cluster and collects the results [12][13].

2.1 MapReduce

MapReduce is a parallel data processing model for substantial data processing on cluster-based computing architectures. A single-threaded implementation of MapReduce is usually not faster than a traditional (non-MapReduce) implementation, but gains are achieved in multi-threaded implementations. Using MapReduce has benefited only when there are fault-tolerance features and the optimized distributed shuffle operation [12][13]. The input data define a MapReduce job; a procedure Map, which for each input element generates several key/value pairs; a phase of shuffle network; reduces a procedure, which receives as input elements with the same key and generates summary information from such elements; the output data. MapReduce guarantees that the same reducer will try all elements with the same key since all the mapper use the same hash function to decide which reducer send the key/value pairs [14][15]. It is based on a master-slave architecture in which a master node handles many slave nodes. At the initial stage, the MapReduce first divide the inputs into equal-sized data blocks for even load distribution. Each block is allotted to a slave node and subsequently processed by a mapper function, and results are generated. The slave node interrupts the master node whenever it is idle. The scheduler then allocates new jobs to the slave node [7][16]. The scheduler considers data locality and resources into consideration during the distribution of the data blocks. The components of the MapReduce architecture are shown in Fig 2.

Job Clients: This component submits the task. This task comprised of the mapper function, reducer function and other configuration function that drives the job.

Job Tracker: The job tracker is the master of task trackers; they execute the work on data nodes. This component comes up with an execution plan, coordinate and schedules it across the task trackers.

Task Tracker: The task tracker divides the task into smaller components, *i.e.,* map and reduce tasks.

Internally, MapReduce has split, map, shuffle and sort, reducer and output phases (*See Fig 3*).

Spilt: In this phase, the input format is used to extract data from the database and divide it into smaller units. By default, the input is of type text which splits the data in the file into the record by record. Each record is split and load to a mapper function. For instance, if the data is an image, the input format is binary. If it is a relational database, the input format is a database.

Map: The mapper picks the data of interest and executes it on the pairs of keys available. It converts the input split into the pairs based on user-defined code.

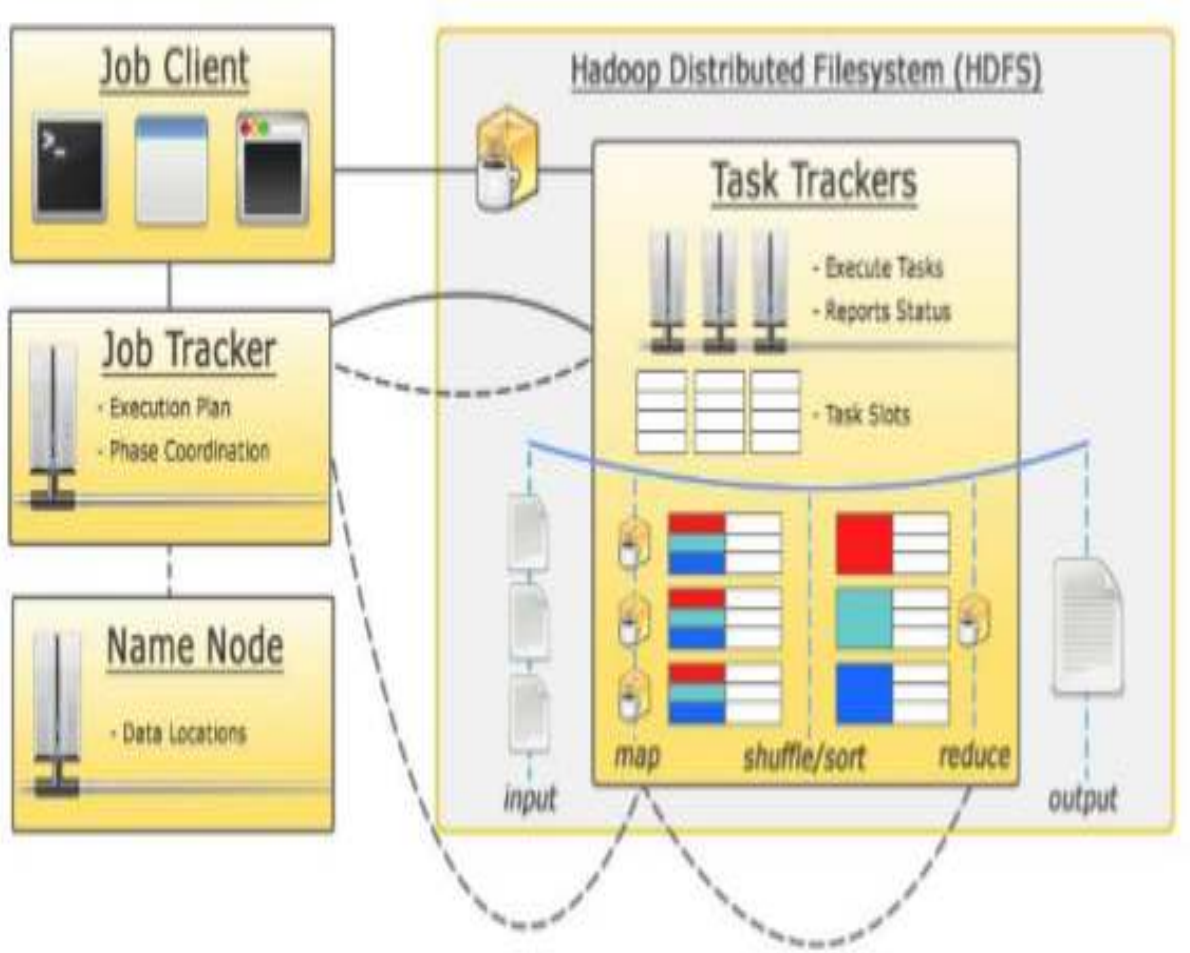

Fig 2: Block View of MapReduce [16]

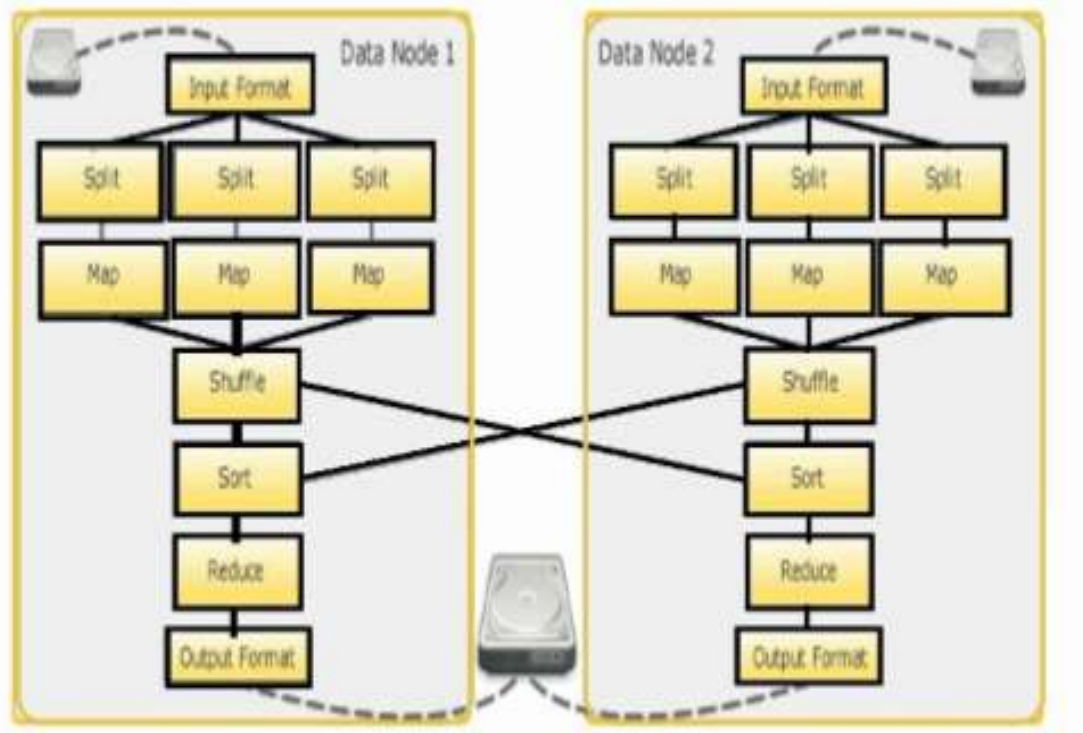

Fig 3: Internals of MapReduce [16]

Shuffle and Sort: This phase picks all the data nodes that are part of the job and group them based on their keys. It sorts and sends it to the reducers.

Reducers: Reducers work on the sorted data and cumulatively combine all the results.

Output Format: This phase presents the result to the user and could also store it in the appropriate storage facility.

2.2 Association Rule Mining

Association Rule Mining (ARM) is a soft computing technique that attempts to find frequent itemsets among large datasets and describes the connection among different attributes. The most typical forms of this techniques are Apriori, Eclat, FP-Growth, and partition. Market basket analysis, text mining, web usage mining, protein sequences and Bioinformatics are the application areas for ARM [17]. Most researchers prefer it because of its non-invasive nature which allows it to handle both quantitative and qualitative data. This has made it to fit well in most real-world applications. ARM is a widespread technique for realizing interesting relations between features and patterns in large repositories [18]. It discovers robust rules in data storage facilities using different procedures of interest. ARM is a rule-based technique in unsupervised machine learning. It shares the common purpose of finding patterns in data that can be given in the form of an IF-THEN rule. Association rule contains particular features of data in the body of the rule that is related to other attributes in the head of the rule [18][19]. The importance of an association rule is measured by two features: support and confidence. Support determines the portion of all records in the repository that satisfy the rule. Rules with higher support are given higher priority. Confidence determines the segment of the records that satisfy the rule. Rules with high confidence have a higher connection between the features described in the head and the features defined in the body. The association mining task can be stated as follows: Let A be a set of items, and T a database of transactions, where each transaction has a unique identifier (*tid*) and contains a set of items. A set of items is also called an *itemset* [20]. The *support* of an itemset X, denoted by $\sigma(X)$, is the number of transactions in which it occurs as a subset. An itemset is *frequent* if its support is more than a user-specified *minimum support (min sup)* value [18]. An *association rule* is an expression $A \rightarrow B$, where A and B are itemsets. The support of the rule is given as $\sigma (A \cup B)$, and the *confidence* as $\sigma(A \cup B)/\sigma(A)$ (*i.e.,* the conditional probability that a transaction contains B, given that it contains A). The mining task consists of two steps: 1) Find all frequent itemsets. 2) Generate high confidence rules. Rules of the form $X \backslash Y \rightarrow Y$ (where $Y \subset X$) are generated for all frequent itemsets X, provided the rules have at least *minimum confidence (min_conf)*.

2.3 Apriori Algorithm

Apriori is the most well-known technique of ARM due to its efficient results in knowledge discovery. It is customarily used for the mining of frequent itemset over databases. It continues by identifying the frequent items in the database and extending them to bigger item sets as long as those itemsets are often seen in the database. These frequent itemsets discovered via Apriori algorithm are used to determine association rules. It employed an iterative technique known as the breadth-first search strategy to count the support of itemsets [20]. The focus is that all nonempty subsets of a frequent itemset must be frequent. There are two main phases in Apriori: prune step - eliminates an itemset if its support is less than minimum support, and abandon the itemset if its subset is not frequent, and join step - candidates are generated by linking the frequent itemsets level wise. The critical setback of this step is that the multiple dataset scans. Suppose there is a transactional database D for a hospital. This hospital wants to analyze the patronage of NHIS patients - by finding the relations between those that often come to the hospital and those that rarely come. This will help the hospital to develop a strategy for handling patients that are under NHIS.

2.4 MapReduce-based Parallel Association Rule Mining Algorithms for Big Data Analytics

The Apriori Association Rules Mining technique is one of the best conventional procedures for uncovering frequent itemsets from a transactional database [21][22][23][24]. Apriori algorithm finds frequent itemset and uses this item to produce rules. These rules are used for detecting unknown relationships, thus, producing outcomes that can be used for decision making.

In handling enormous datasets using the Apriori algorithm, certain issues such as high memory usage and computational cost can be very prevalent [5][6][7][17]. Similarly, a single processor's memory and central processing unit resources are limited, which accounts for the inability of this technique to perform efficiently. One way to improve the performance and efficiency of this technique (Apriori algorithm) is by parallelizing and distributing the process of generating frequent itemsets and association rules [22][24]. Parallelizing and distributing the Apriori algorithm would improve the performance of the mining process. However, this is also prone to some issues such as workload balancing, partitioning of input data, reduction of the communication costs and aggregation of information at local nodes to form the global information. Other problems prevailing in most distributed frameworks such as the overheads of managing the distributed system and the lack of a high-level parallel programming language are also likely to occur. Again, working with a large number of computing nodes in a cluster or grid, there is always the potential of node failures, which cause multiple re-executions of tasks [16][23]. All these drawbacks can be tackled using the MapReduce Framework, which was introduced by Google [24].

The MapReduce Framework, as discussed earlier, is a Java-based software development model for readily and efficiently building applications that analyze enormous datasets in parallel on large clusters of commodity hardware in a trustworthy failure resilient manner. The user specifies 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 [23][24].

Association rule mining identifies rules that describe a portion of large data to discover useful insight from it [20][22][23]. Implementing it with MapReduce framework has gone a long way in addressing big data analytics problems. For instance, Oweis *et al.* [25], developed a MapReduce model using lift association rule mining algorithm for the processing of big data. This was used to uncover some hidden knowledge and patterns from immense, complex, and multi-dimensional datasets. The conventional ARM methods cannot handle this enormous data available today; hence, implementing the algorithm with the MapReduce framework enabled it to handle large datasets with a large number of nodes. This work was very efficient in measuring the correlations between itemsets. The work of Singh and Miri [26] developed an algorithm using Apriori algorithm in parallel implementation based on MapReduce Framework that improves the processing time for the second iteration in frequent itemsets mining. This algorithm easily handles a large amount of data for mining with less processing time, which eased a big data analytics task.

Finding frequent itemsets is one of the most critical features in big data mining, and Apriori algorithm is the most established algorithm for finding frequent itemsets from a transactional dataset; however, it needs to scan the dataset many times and to generate many candidate itemsets. Ramteke [27], developed an efficient parallelized Apriori algorithm using the MapReduce framework, which needs only two phases (MapReduce Jobs) to extract all frequent itemsets. This algorithm outperformed most existing algorithms. Similarly, Nancy *et al.* [28], harnessed the parallel nature of MapReduce and combined it with Association Rule Mining to build an analytic tool for the processing of big data. Association rule mining is very vital in uncovering hidden patterns from the dataset, but when the volume of the data becomes excessively high, generating rules at a faster pace seems impossible. By applying the parallel execution in the MapReduce framework, the rules can be generated much faster and in an efficient way. This algorithm converted the input dataset into key-value pairs in MapReduce and parallelized all the stages of association rule mining algorithm as well.

Prajapati *et al.* [29], studied the intriguing association rule mining with both consistent and inconsistent rule discovery from massive sales data in a distributed setting. The procedures of processing such massive dataset were computationally complex when using conventional techniques. In the research, a new method was developed to discover consistent and inconsistent association rules from enormous sales data located in a highly distributed environment to curtailed the issues with main memory and computing time of single computing systems by applying computations to multiple nodes clusters. This extracts frequent itemsets from each node using existing distributed frequent pattern mining algorithms. Thus, MapReduce based consistent and inconsistent rule detection algorithm detects the consistent and inconsistent rules from big data and provide useful and actionable knowledge to the domain experts. These pruned interesting rules also give useful knowledge for better marketing strategy as well. The extracted consistent and inconsistent rules were evaluated and compared based on different interestingness measures presented together with experimental results that lead to the conclusions.

With the enormous amount of data available, especially in the healthcare industry, traditional methods of processing data are time-consuming and inefficient due to the complicated nature of medical processes and the complexity of the data. With this, a favourable niche has been created for criminal in this sector as most data remain unanalyzed for an extended period [30]. Conventional analysis methods are not suitable due to their inherent limitations to manage the volume, velocity, variety, veracity of the data in healthcare. In an attempt to address this problem, Devi and Sarojini [21], considered the applications of association rule mining to analyze data from different databases. This was done via the use of expert methods and techniques to recognize trends and profiles hidden in data. This aided in the processing of large and distributed databases - logistics, marketing and Government - almost all branches, *e.g.* defence, public safety, spatial database - GIS, relational database - industries, medical database- medical diagnosis, hospital, medical shops, scan centres.

Deshpande and Anami [32] developed a system for the analysis of arthritis patients' data using Association Rules Based. Data mining in health care is also known as predictive analysis and has been an area for research. The research involves the implementation of association rules using Apriori Algorithm for depicting the co-occurrences between particular arthritis and their factors. This contributed more in the quick detection of arthritis and its factors. Similarly, Kang'ethe and Wagacha [33], developed a system for mining diagnosis features in electronic medical records using association rule mining. This research

disclosed that association rule mining could not only be applied to confirm what is already in existence from health data in the form of comorbidity patterns but also generate some exciting and strong disease diagnoses associations that could provide a good starting point and room for further exploration through studies by medical researchers to explain the patterns that are seemingly unknown or peculiar in the concerned populations.

Al-Maolegi [34] indicate the limitations of the original Apriori algorithm of wasting time for scanning the whole database searching on the frequent itemsets and presented an improved Apriori algorithm that addressed the time waste during scanning of the database. The research showed experimental results that used the original Apriori and the improved Apriori. This shows that the new Apriori reduces the time consumed by 67.38% in comparison with the original Apriori, and makes the Apriori algorithm more efficient and less time consuming. The results show that the improved Apriori algorithm that scans only some transactions instead of the whole database reduces the consumed time. Similarly, Rao [35] presented a new scheme for finding the rules out of transactional datasets which improve the original Apriori in terms of the number of database scans, memory consumption, and the interestingness of the rules. It also avoids scanning the database again and again.

Woo [36] presented an Apriori algorithm that runs on parallel MapReduce framework, namely Apache Hadoop over a cluster of computers. The author also discussed the time complexity, which theoretically shows that the algorithm has a higher performance than the sequential algorithm. The item sets produced by the algorithm can be used to compute and produce association rules for market analysis. Mahendra [37] developed an algorithm to mine the data from the cloud using sector/sphere framework with association rules. The researcher also discussed the integration of Sector/Sphere framework and Association rule. This enables the application of association rule algorithm to the wide range of cloud services available on the web. Sphere allows developers to write certain distributed data parallel applications with several simple APIs. A Sphere database consists of one or more physical files. A user-defined function does computation in the sphere. The result could be written to either the local disk or common destination files on other nodes.

Hegazy [38] implements an efficient MapReduce Apriori algorithm based on Hadoop-MapReduce model which needs only two phases to find all frequent itemsets, the researcher also compared the new algorithm with two existed algorithms which need either one or more phases to find the same frequent itemsets. Experimental results showed that the proposed Apriori algorithm was efficient and exceeded the other two algorithms.

Dean [39] developed a MapReduce system that runs on a large cluster of commodity machines and was highly scalable. The author added some optimizations in the implemented of the system that aims to reduce the amount of data sent across the network. The system gave a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs.

Leem [40] developed a system to assist the database and open source communities in understanding various technical aspects of the MapReduce framework. The research discussed the MapReduce framework's pros and cons. Also, the researcher introduced the optimization strategies and discussed challenges raised on parallel data analysis with MapReduce. It found that MapReduce is simple but provides excellent scalability and fault-tolerance for massive data processing.

Yang [41] improved the MapReduce framework by adding a Merge phase so that it is more efficient and easier to process data relationships among heterogeneous datasets. Also, the researchers extended the MapReduce framework to the Map-Reduce-Merge framework. It also adds a new Merge phase that can join reduced outputs. The researcher found that Map-Reduce-Merge model added to MapReduce's many features. It also contains several configurable components that enable many data-processing patterns.

2.5 MapReduce and NoSQL in Big Data Analytics

There are so many approaches in NoSQL that focus on the management of different varying data formats (structured, semi-structured and non-structured) and with the motive to address particular issues [42]. Many researchers (data scientists, researchers and business analysts) pay more attention to the agile approach that leads to prior insights into the datasets, which may be masked or constrained with a more formal development process. The NoSQL databases are open source in nature, horizontal scalability, easy to use, store complex data types, swift for adding new data and for simple operations/queries. Some commonly used NoSQL databases are MongoDB and Apache Cassandra [43][44][45].

According to a survey carried out by Pothuganti [44], there are possibilities of combining MapReduce and NoSQL Databases to build an analytical tool for big data. The researcher considered exponential growth of data in the universe, which was likened to oxygen in the universe. It challenges to cutting - edge businesses such as Google, Yahoo, Amazon, Microsoft, Facebook, Twitter *etc.* were presented in the work. The growth of this unstructured, semi-structured and structured data that gets created from various applications, such as emails, weblogs, social media demands new strategies for processing and analyzing information because it has exceeded the processing capacity of conventional database systems. To this end, big data analytics techniques such as Hadoop MapReduce and NoSQL Database have been widely employed to address analytics issues in big data.

Moreover, Maitrey and Jha [42] developed a simplified big data analysis system using MapReduce to curtail the problem of the continuous increase in the growth of data with the increasing need of data processing which is getting arisen from every scientific field. A big problem has been encountered in various fields for making the full use of these large-scale data which support decision making. The researchers used data mining techniques to discover new patterns from large data sets. MapReduce was used to implement the data analysis model for information retrieval field.

Ananth and Raghuveer [46], developed a novel approach using MongoDB for big data analytics. In their view, everything around the universal is data, and everything in the universal generates many data. A smartphone, for instance, generates loads of data daily: phone logs, message logs, mail data *etc*. There are other gigantic devices today and just imagine each device churned out loads of such data! Likewise, there are more than a trillion websites today (there cannot be any more websites that could be created with the IPv4 protocol shortly). Each website puts up data in the form of ZetaBytes or even more. Facebook *per se* handles more than 30+ Petabytes of user data. What seems to be small today would be big tomorrow. It

is no longer a trend but rather, a boon and exponential growth of data. Storage of such data and processing is always a problem.

3. MATERIALS AND METHODS

This section described the methods deployed in this research.

3.1 Case Study

NHIS was set up by the Federal Government of Nigeria to provide universal access to quality healthcare service in the country. It covers civil servants, the armed forces, the police, the organized private sector, students in tertiary institutions, the self-employed, vulnerable persons, and the unemployed, among others (NHIS, 2013). The beneficiaries are required to pay a premium to NHIS, which is used to pay for their healthcare services once they visit the facility. The primary purpose of the analytics of health insurance data is to investigate the cost of healthcare services (*i.e.,* payment made for healthcare services by health insurance). Hence, this research aims at exploring the huge data available for increasing the effectiveness and efficiency of the scheme [2][7][45][46][47][48].

The inability to sufficiently derived insights from big data has led to a series of problems in organizations resulting in the provision of poor quality services to their clients. These problems range from corruption to the lack of appropriate data analytics tools for the processing of the data available. As a major problem in developing countries, corruption has played a significant role in diminishing the fortunes of many organizations. For instance, many organizations often engage the services of staffers who are not qualified because of either personal relationships or pecuniary attractions; also, resources designated for the operation of such organizations often times find vent in the private custody of the managers, thus leading to many challenges that account for its operational sluggishness [49][50][51]. A precursor to this high rate of corruption in the healthcare insurance industry, for instance, is the use of manual data analytics methods in the processing of the data resulting from its daily operations. These have led to critical delays in the analysis of such data which is continuously collected from different sources and formats via registration, update complaints/inquiries, claims, referral, subsidy gap funding, monitoring and quality assurance [49][50][51][52][53][54][55]. This has leftover 80% of the data un-analyzed, which is the reason for the non-availability or late delivery of the required information for decision-making [52][53][54][55]. Extant studies [54][55] focused on the automation of the process of capturing data into structured databases with little or no attention given to the processing of this data. Forcing this unstructured data into structured databases has led to the loss of some vital data required during data processing. This is responsible for the delay in the processing of enrollment, referral and claim data. However, deriving business outcomes via more in-depth insight from this data is a competitive necessity. It requires a new approach to combine and correlate the data.

3.2 Data Collection

The NHIS data were collected from various Health Maintenance Organisations (HMOs) and stored into MongoDB in its raw form. Collecting this data was one of the most difficult tasks in this work because most of the data were collected manually by NHIS through HMOs and stored in file cabinets. This was done via document examination and observation, which in either case, the data was collected from journals and NHIS databases. Data in hardcopy form was digitized by scanning it into the computer system before use. The documents are scanned as images either in .jpg, .png and .gif or pdf formats. Some healthcare facilities submit the data in pdf, excel, csv, docx or txt formats and some as scanned images which do not require further digitization. Categories and features of the data are shown in Table 1. Over ten million records were collected to identify the current challenges bedevilling the existing methods employed. Existing data was extracted using Optical Character Recognition tools, and model parameters identified.

After collecting the data, it was pre-processed. This stage converts the different formats of data in the MongoDB into a single format. In this case, the data was converted into comma-separated values (CSV). This has three (3) steps: the data is extracted from the MongoDB, then converted into CSV, and the data in the CSV file is partitioned into individual fields and passed to the map function. Here, the data is retrieved from MongoDB and features of interest are extracted with the aid of the Optical Character Recognition (OCR) library, which is implemented in the system. These features are stored in a temporary CSV file as an intermediate output of the pre-processing.

3.3 Distributed Apriori algorithm for MapReduce

Building the distributed Apriori is the core of this research. As mentioned earlier in Section 2, the Apriori algorithm is a technique that is used to find frequent itemsets in the transactional database. To implement Apriori with MapReduce, there is a need to understand how it works:

1. The first step is to find L_1, which counts all the item in the database (in this case, it will count all the items in X (data of interest) after extraction).

   This step is very straightforward in MapReduce.

a. In Map Task, each document will input to the map (Key, Value) function. The key is items, and the value is one. The Divide-and-Conquer Approach is applied to X at this point.

b. The Reduce Task, after shuffling and sorting the data, is based on keys. This phase used the principle of the set intersection to collate similar items together using the *Reduce (Key, Value [])* function. The key is the item, and the value is a list of counts of that item. The Reduce task will iterate over the value and output of the final sum.

2. When the level is 2, the join is required to find the next level. This can be done in the Map task. Reduce task will find the final sum of the items and filter them based on minimum support.

3. It is stopping criteria if L_k is empty. This can be done quickly by checking the stop flag using an if-statement.

Table 1: Data Categories

Data Category	Description
Enrolment	**Provider:** Name, Address, Telephone, Fax/Phone, Email, Type of facility, Category of registration, State registration no, Name of Director, Name of supervising Medical Director (If applicable), Affiliated HMOs, Affiliated Insurance companies, NHIS registration number, Incorporation/business registration.
	Beneficiary: Name, Address, Date of birth, Sex, Next of Kin, Email address, Mobile, Telephone no. fixed, National ID no, Employer NHIS no., Date of NHIS registration, Nationality, Location of Posting, Photograph, Blood group, Genotype, Allergies, Relationship (Principal, Spouse, Child, Extra-dependant), Expiry date, Primary provider
Payment	**Claim:** Name, NHIS No. of patient, Name and NHIS No. of patient's primary provider, Name and NHIS No. of Secondary Provider, drug prescription sheet, Diagnosis/disease code, Treatment given, Date of treatment, Amount billed, Co-payment received (when applicable).
Update	Addition of dependents, change of facility, change of HMO
Referrals	Referral request, approvals, rejections

Implementing this involves five main phases: pre-processing phase, generating frequent itemset for each split, generating frequent itemset for the entire data, generate association rules phase, and the output phase. Fig 4 shows the workflow of the proposed model, while Fig 5 shows the detailed representation of the proposed model. As a MapReduce processing model, in the first phase (*i.e.,* the pre-processing phase), three (3) steps are conducted (i) the data of interest, X, is extracted from the MongoDB (ii) this data is converted into a CSV with m fields (iii) the data in the CSV file is partitioned into individual fields and passed to the map function. Each map function takes one split as input, and there are also a mapper and reducer functions. The output of this phase is a constant k- itemsets and their occurrence for each split as a list of intermediate key/values. The second phase has a MapReduce computation for generating frequent candidate itemsets for all the data. The input of this process is an input split, and a file containing all frequent partial k-itemsets that resulted from the first phase. The output is the frequent k-itemsets and its occurrence in the whole input data. Finally, the association rules are generated using frequent itemsets.

As a MapReduce processing model, in the first phase (*i.e.,* the pre-processing phase), three (3) steps are conducted (i) the data of interest, X, is extracted from the MongoDB (ii) this data is converted into a CSV with m fields (iii) the data in the CSV file is partitioned into individual fields and passed to the map function. Each map function takes one split as input, and there are also a mapper and reducer functions.

Data collection: The NHIS data is collected from various HMOs and stored into MongoDB in its raw form. Collecting the NHIS data is one of the most difficult tasks in this work because this data is collected manually by NHIS through HMOs and stored in file cabinets. This requires digitizing the data by scanning it into the computer system before anything can be done on it. The data was collected from different HMOs whose identities have been concealed for obvious reasons.

Pre-processing: Here, the data is retrieved from MongoDB and features of interest are extracted using OCR. These features are stored in a temporary CSV file as an intermediate form of the pre-processing. Applying the Apriori Approach usually requires a pre-processing stage that would convert the different formats of data in the MongoDB into a single format. In this case, the data is converted into the comma separated values (CSV). This is achieved by using the OCR to extract all the fields in the data. The algorithm for this phase is shown in Algorithm 1.

Loading: The features stored in the CSV are split into individual features and executed in parallel by the MapReduce. The algorithm for this phase is shown in Algorithm 2.

Generate frequent itemsets and their occurrence in the split: In this step, the itemsets are generated for the split resulted from the previous step, and the MapReduce model outputs the itemsets along with their occurrences in the split using a Map and Reduce function. In this phase, the data is divided into logical Input Splits, each of which is then assigned to a Map task then the map worker calls the map function to process the input split. The Map function shown in Algorithm 3 reads one split at a time and output a list of intermediate (key, values) pairs where the key is the element of the attribute set, and the value is its occurrence. The data from the mapper is passed to the reducer (see Algorithm 4) which takes it and get all the values associated with the same key and writes the value in the output file in the increasing order of the keys. The output is a list L of (key, value) pairs where the key is an element of attribute set and the equal value one, and this list Li is stored in a temporary file.

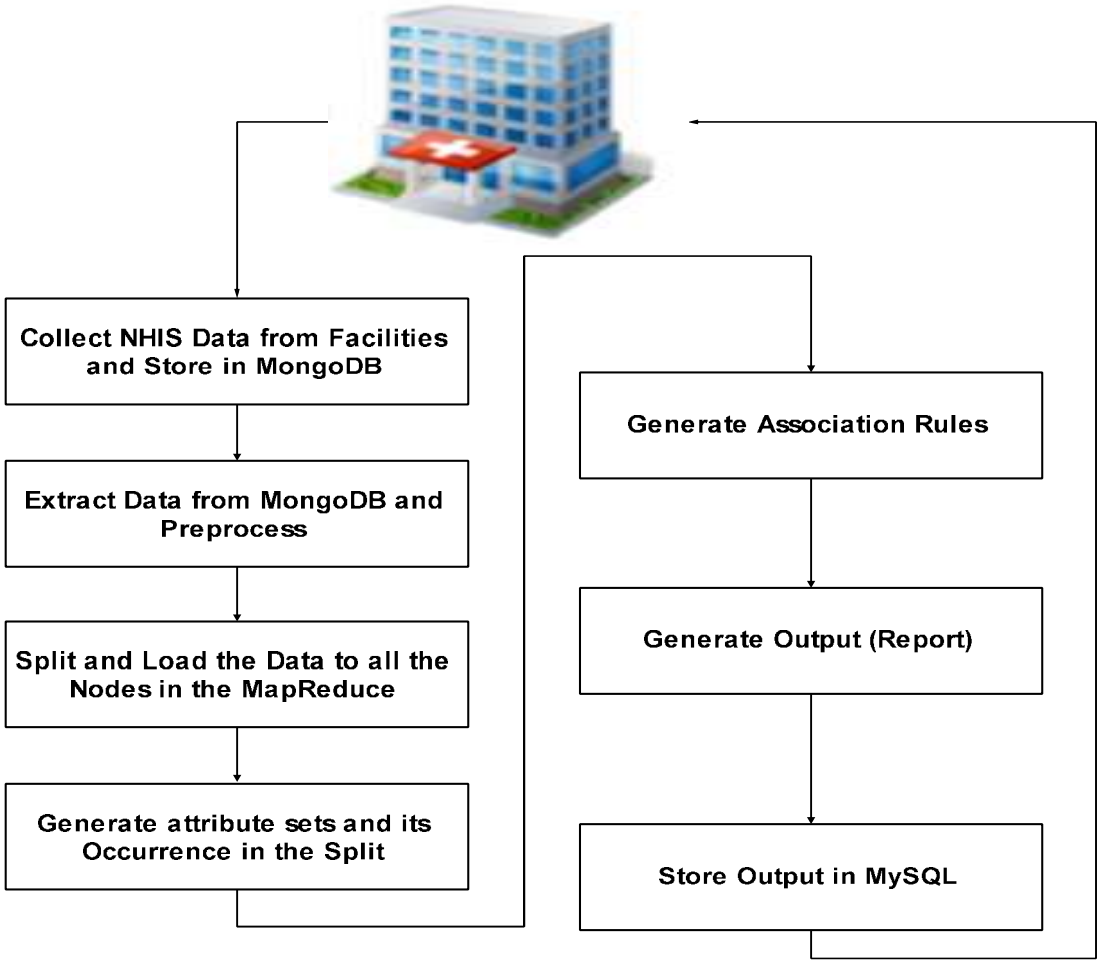

Fig 4: Workflow for BDAM4N

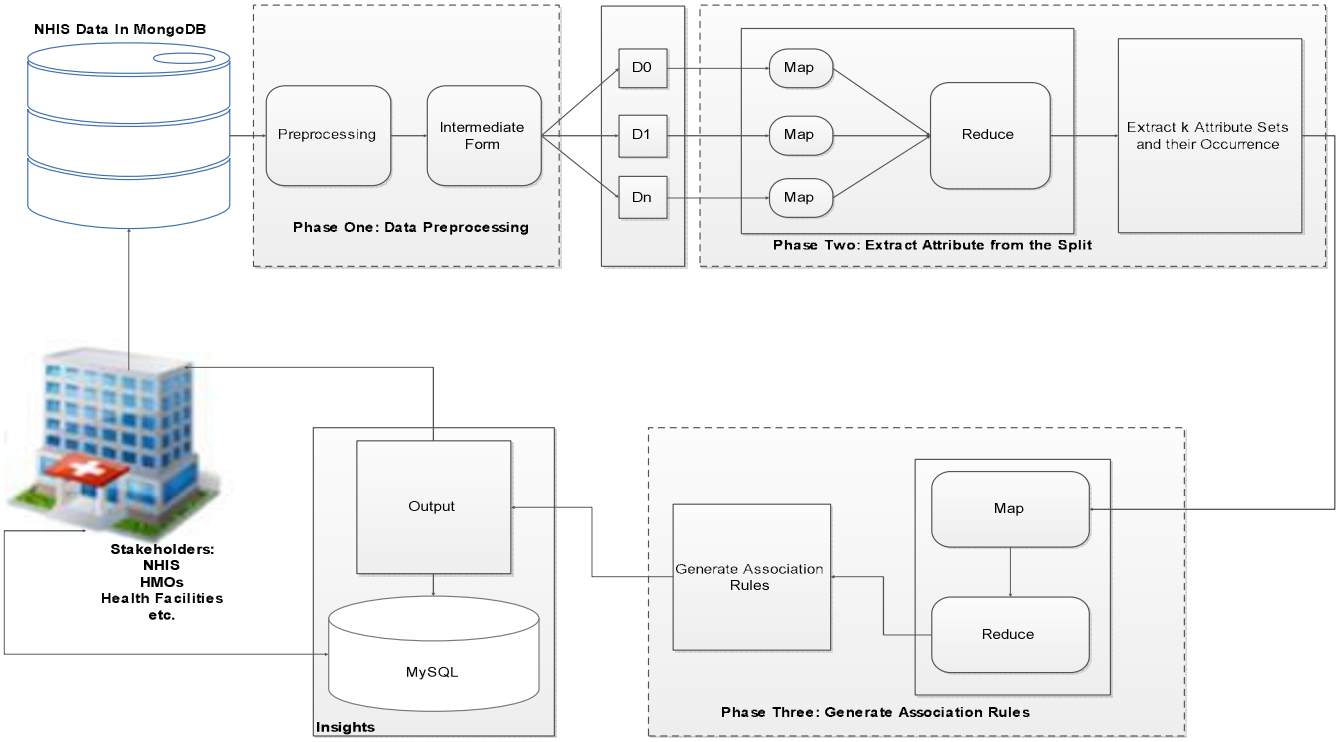

Fig 5: Detailed Representation of BDAM4N

Algorithm 1: Data Preprocessing

Input: D the data in the MongoDB database.
Output: csvfile
Read documents from D.
Value: ocred_content, // content of the document after applying OCR
ocred_content = OCR(content)
split D into k partitions //combinatorial principle from Ramsey Theory
extract docnam; // apply the principle of set intersection to extract this document's name
extract content; //apply the principle of set intersection to extract the content of this document
For each line ∈ ocred_content
feature=extract (line); // extracting features from documents
insert comma (,); // insert a comma after extracting the features
If (ocred_content.hasMore.Features)
docFeat ← docFeat ∪features;
end if
end for
Context.write(csvfile, docFeat);
END

Algorithm 2: Data Loading

Input: CSVfile.
Output: features
Read features from CSVfile.
For each line ∈ features
If (line.has.comma)
feature=extract (line); // extracting features from documents
splitFeat ← feature;
end if
end for
map(splitFeat);
END

Algorithm 3: Map Phase

Input: split of the data, Si
Output: (key, value),
key: element of k-itemset,
value: the occurrence of the element,
Map(object) // Map function
L1=separate attributes of each beneficiary from Si
For(k=2; $L_{k-1} \neq \emptyset$,;k ++)
Generate new candidate C_k;
For each candidate c ∈C_k c.count++;

Sort all the attributes with same key and order them by their unique number

end for

L={c∈C_k }

if itemset I ∈ L

output (I , c);

end map

END

Algorithm 4: Reduce Phase

Input: (key1, value1),

Key1: element of k-itemset,

Value1: the occurrence in the split,

Output: (key2, 1),

Key2: element of frequent candidate k-itemsets,

Reduce(key1, value1) // Reduce function

Out(key2, 1); // collected in L1

End reduce

End

Generate association rules: After generating the frequent itemsets for all the data, the association rules generated using Map and Reduce functions with predefined minimum support and confidence to generate strict rules. This is shown in Algorithm 5. The Map function takes the list of frequent itemsets and their support and takes the frequent itemset that survived the support threshold and group the entries of the same key. The Reduce function calculate the confidence of each itemset and output the itemsets which satisfy the confidence threshold. The output will be the rule with confidence.

Algorithm 5: Map and Reduce function for Rules Generation

Input: a set of all attributes, Va = {$a_{1\,U}$ U a_2 U…Ua_n}

Output: set of rules

Map Function (Va)

for each attribute a, do

for each correspondence object x, do

construct the decision rule ($c_1 = v_1 \cap c_2 = v_2 \cap \ldots \cap c_n = v_n$) ➔ du = u

End for

Reduce Function(u):

//For the enrollment of principal beneficiary, scan for the existence of the records

Scan the u over an object x // name, date of birth, gender, phone number, email

IF x does not exist THEN

enroll x

***else** x is already in existence //Double registration*

//For enrollment of spouse

If NHIS No of the principal is valid, THEN

IF the marital status of the principal is Married, THEN

If the residential address of the spouse and the principal is not the same, THEN

Enrol Spouse with a different Health Facility

Else

Enrol Spouse with same Health Facility as the principal

Else

REJECT Spouse Enrollment, Principal not yet married

Else

REJECT Spouse Enrollment, and the principal does not exist

END

//For enrollment of a child

If NHIS No is valid, THEN

IF the marital status of the principal is Married, THEN

If the number of children enrolled so far is not up to four, THEN

If child/children age (s) is not more than eighteen, THEN

If the address (es) of the child (ren) is not the same as the principal, THEN

Enrol Child (ren) with a different primary Health Facility

Else

Enrol Child with same Health Facility as the principal

Else

REJECT child (ren), Age (s) is/are above 18years

Else

REJECT child (ren), the number of children is more than four for this enrollee

Else

REJECT, not yet married

Else

REJECT, the principal does not exist

END

//For the enrollment of Extra dependant

If NHIS No of principal is valid, THEN

If extra dependant for the principal is not up to one, THEN

If the extra dependant is not in the same location as the principal, then

Enrol dependants with the different primary provider

Else

Enrol Dependants with the same provider with the principal,

Else

REJECT, the number of allowable dependants is exceeded

Else

REJECT, the principal does not exist

END

//Update (change of facility

If NHIS No is valid, THEN

If the last date update was carried out is up to six months, THEN

Update facility for the enrollee

Else

REJECT, the last update was done less than 6months ago

Else

REJECT, Principal does not exist

END

//Note: update applies the same rules as in the enrollment

//Referral

IF NHIS No is valid THEN

Extract the doctor's diagnoses for the enrollee, dd

Extract NHIS approved diagnoses, nd

Extract the details of the facility they are referring to, rh

Extract details of the primary facility as well, ph

IF dd ∈ nd THEN

IF rh==Secondary facility OR Tertiary OR rh specializes in cases like dd, THEN

Compute the distance d between ph and rh

Also, compute distance, fd, of other facilities in the same area with ph

IF d is less than OR equals fd, THEN

Approve the referral request

Else

Reject the referral request //Traces of Fraud observed

Else

Set approval status to 'REJECTED' //Traces of Fraud observed

END

Construct (ci, 1 ≤ i ≤ n)

For every c ∈ C do

Assign the value V to the corresponding attribute a

End for

Construct a decision attribute d

Assign the value u to the decision attribute d

Assign the value u to the corresponding decision attribute d

End for

End for

End

Report: An output is generated at this stage in the form of a report. This report shows whether a particular data category is marred with irregularities or not, *e.g.,* a claim contains irregularities or free of irregularities.

Storage: The output is then stored in a relational database where it can be accessed by the stakeholders either immediately or anytime the need arises. The relational database used here is the MySQL database.

4. RESULTS

The prototype of the system was implemented using Java Programming Language, Apriori algorithm as the machine learning algorithm, MongoDB as the big data storage

mechanism, and MapReduce as the big data processing platform. The implementation of the system has two folds: a standalone data capturing system which was built to ease the data capturing processing since none of the data capturing systems developed for the scheme has not been implemented. (The prototype tool was developed using Java Enterprise Edition technology. It is web-based and can be used in a distributed setting. The front-end is built using Java Server Pages (JSP 2.3), the business logic is implemented using Enterprise Java Beans (EJB 3.2), and the back-end is implemented in two folds: input storage was implemented using MongoDB, and the output storage was implemented using Structured Query Language (SQL-MySQL). The input storage (MongoDB) accepts data in different formats (*e.g.,* pdf, jpeg, png, gif, csv and excel) as inputs and process them to produce a report for the stakeholders which is stored in MySQL. The considerable input in the different formats is first divided into smaller units using the Divide-and-Conquer Algorithm, which is implemented in the MapReduce Framework to ease the processing of this data.

4.1 Experimental Design

In order to ascertain the response time and throughput of the system, two sets of four experiments were carried out. The first set of experiments were carried out with a single node in the MapReduce. In the second set of experiments, multiple nodes were used in the MapReduce. These number of nodes were varied according to the increment in the records under consideration. In the four experiments, the data was partitioned into four: 71543, 115427, 279950 and 396428, with the increment factor as 0%, 61%, 143% and 42% respectively. With this, the MapReduce nodes in the second set of experiments were 9, 15, 36 and 51, respectively. Table 2 shows details of the variation in the MapReduce framework in the rule generation phase of the system.

Table 2: Variation of MapReduce Nodes

NRs	Increment in Data (%)	Number of Nodes
71543	0	9
115427	61	15
279950	143	36
396428	42	51

Abbreviations used in the Evaluation

Response Time

NU: Number of Users

CRTE: Response Time for the processing of Enrollment data at Constant number of nodes

CRTC: Response Time for the processing of Claims data at Constant number of nodes

CRTR: Response Time for the processing of Referral data at Constant number of nodes

CRTU: Response Time for the processing of Update data at Constant number of nodes

VRTE: Response Time for the processing of Enrollment data at Varying number of nodes

VRTC: Response Time for the processing of Claims data at Varying number of nodes

VRTR: Response Time for the processing of Referral data at Varying number of nodes

VRTU: Response Time for the processing of Update data at Varying number of nodes

CART: Average Response Time for the processing of data at Constant number of nodes

VART: Average Response Time for the processing of data at Varying number of nodes

Throughput

NR: Number of Records

CTPE: Throughput for the processing of Enrollment data at Constant number of nodes

CTPC: Throughput for the processing of Claims data at Constant number of nodes

CTPR: Throughput for the processing of Referrals data at Constant number of nodes

CTPU: Throughput for the processing of Update data at Constant number of nodes

VTPE: Throughput for the processing of Enrollment data at Varying number of nodes

VTPC: Throughput for the processing of Claims data at Varying number of nodes

VTPR: Throughput for the processing of Referrals data at Varying number of nodes

VTPU: Throughput for the processing of Update data at Varying number of nodes

CATP: Average Throughput for the processing of data at Constant number of nodes

VATP: Average Throughput for the processing of data at Varying number of nodes

Result Summary

CE: Enrollment data processing at Constant number of nodes

CC: Claims data processing at Constant number of nodes

CR: Referral data processing at Constant number of nodes

CU: Update data processing at Constant number of nodes

CA: Average data processing at Constant number of nodes

VE: Enrollment data processing at Varying number of nodes

VC: Claims data processing at Varying number of nodes

VR: Referrals data processing at Varying number of nodes

VU: Update data processing at Varying number of nodes

VA: Average data processing at Varying number of nodes

4.2 Response Time

Literary speaking, response time is the time from the moment a user sends a request until the time the application indicates that the request has been completed. In establishing the response time of the model, the response time for all the categories of data (see Table 1) was tested. This was carried out four times

using different sizes of data (*see Table 2*). The outcome of the testing is presented in Table 3a and Table 3b; the respective graphs are shown in Fig 6a and Fig 6b. Here, there was a great significant difference in the response time of the two experiments.

Table 3a: Response Time for the Processing of the Data

NU	CRTE	CRTC	CRTR	CRTU	VRTE	VRTC	VRTR	VRTU
5	0.80	0.80	0.80	0.80	0.610	0.190	0.390	0.470
7	0.85	0.85	0.86	0.86	0.464	0.107	0.227	0.297
9	0.89	0.88	0.89	0.89	0.398	0.069	0.169	0.199
12	0.91	0.91	1.00	1.00	0.293	0.007	0.110	0.020

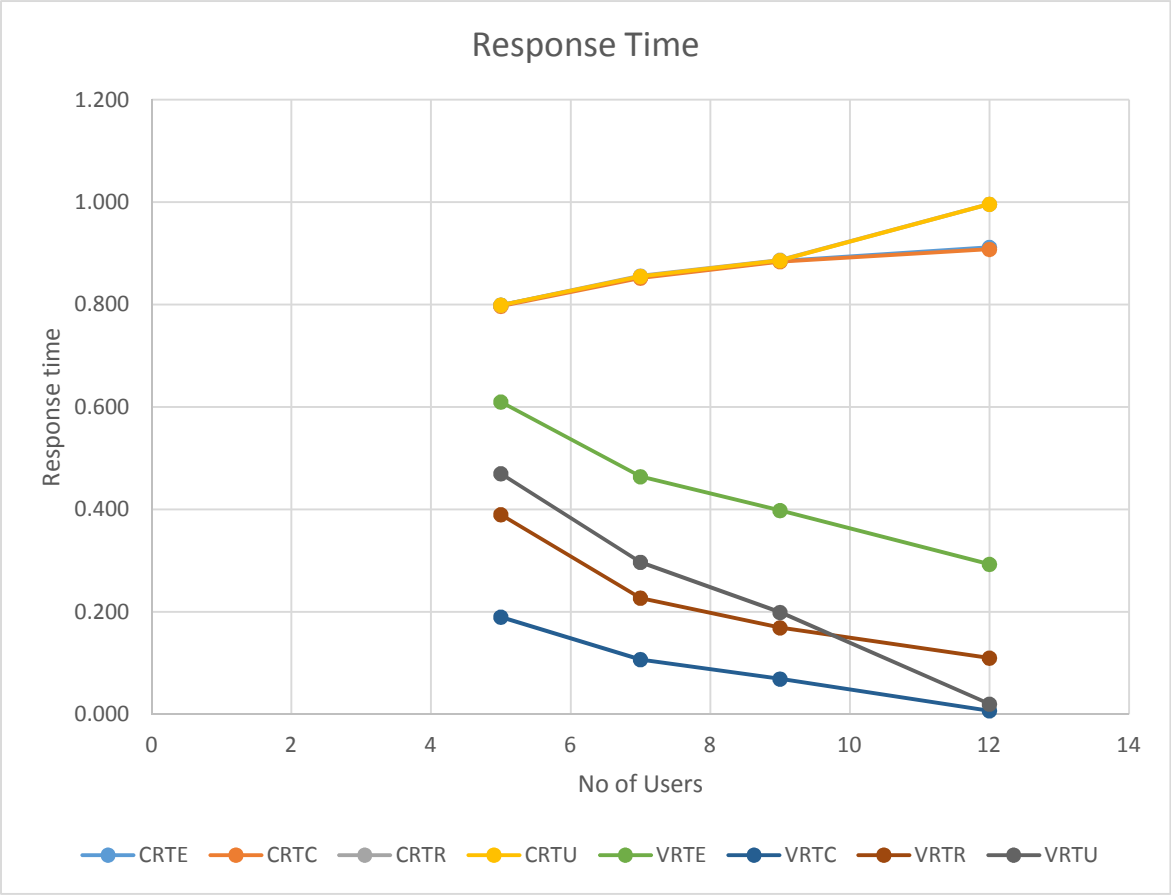

Fig 6a: Graph for the Response Time

Table 3b: Average Response Time for the Model

Category of Data	Average Data per User	CART	VART
Enrollment	23735.25	0.86	0.44
Claims	23735.25	0.86	0.09
Referrals	23735.25	0.89	0.22
Update	23735.25	0.89	0.25
Average	23735.25	0.56	0.25

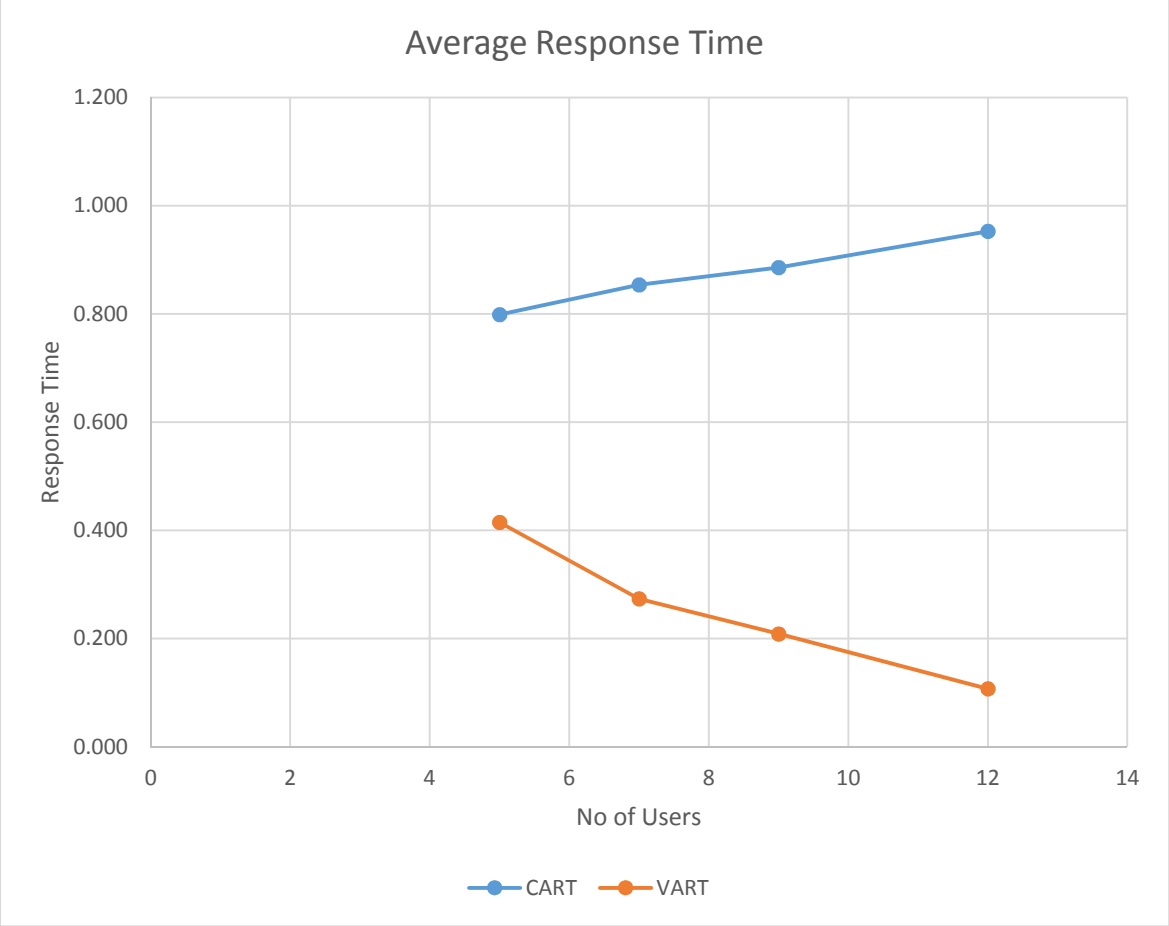

Fig 6b: Graph for the Average Response Time for the Model

4.3 Throughput

This is the volume of data processed in a given time. One of the main characteristics of big data is volume. The volume is always outrageous, and with high velocity, it is always difficult processing it most, especially when it is manually done as in the case of NHIS. Because of the parallel framework of the MapReduce, which is used in the implementation of this model, the throughput is quite high. In establishing the throughput of the model, all the categories of data (*see Table 1*) were tested. This was carried out four times using different sizes of data (*see Table 2*). The outcome of the testing is presented in Table 4a and Table 4b; the respective graphs are shown in Fig 7a and Fig 7b. The throughput for the model increases as the volume of data increases in both experiments, as shown in Table 4a and Table 4b, and Fig 7a and Fig 7b. The average throughput for the model for the first set of experiments is 2796.53 records per seconds. In contrast, in the second set of experiments, it was 8865.29 records per seconds which means that the model can process this amount of data in one second irrespective of the formats.

Table 4a: Throughput for the Processing of the Data

NR	CTPE	CTPC	CTPR	CTPU	VTPE	VTPC	VTPR	VTPU
71543.00	1172.84	831.90	1882.71	2044.09	3109.21	4354.41	5507.54	4930.6
115427.00	1522.78	769.51	2748.26	2493.02	4140.14	5333.96	6695.3	6660.53
279950.00	3313.02	1349.16	4868.70	4761.05	8870.41	11384.71	11907.7	13426.86
396428.00	3450.20	1396.36	5838.41	6302.51	10722.96	12821.09	14720.68	17258.51

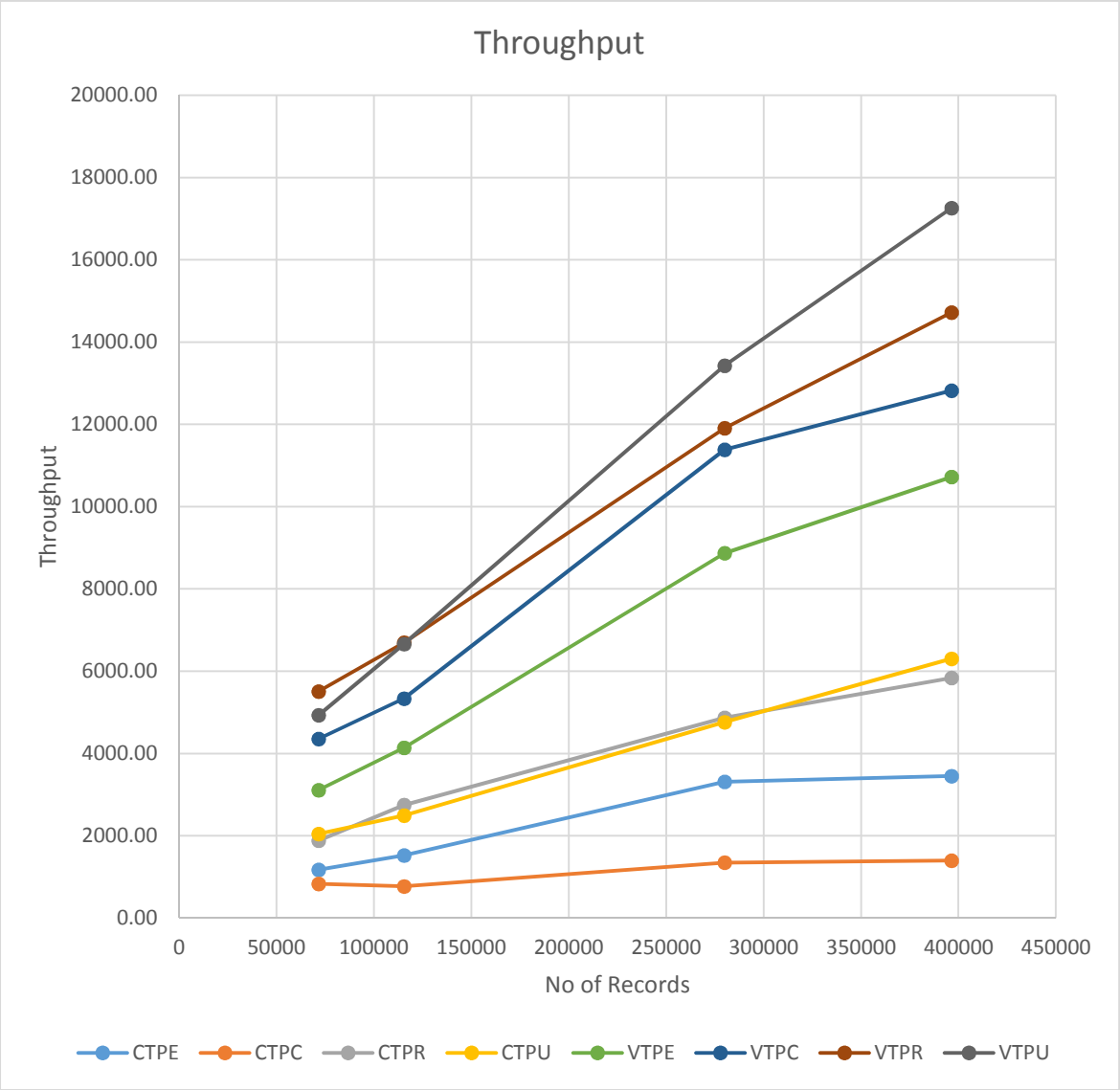

Fig 7a: Graph of the Throughput for the Processing of NHIS Data

Table 4b: Average Throughput for the Model

NR	CATP	VATP
71543.00	1482.89	4475.44
115427.00	1883.39	5707.48
279950.00	3572.98	11397.42
396428.00	4246.87	13880.81

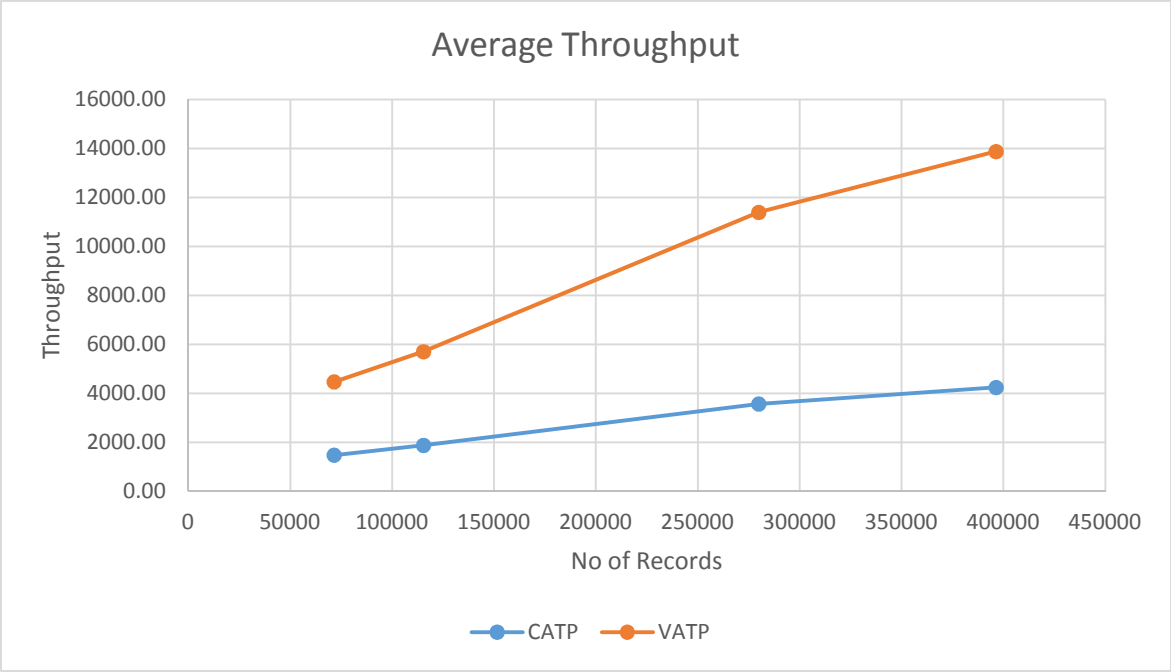

Fig 7b: Graph for the Average Throughput for the Model

4.4 Stability

Confidence is an important measure in determining the reliability of a rule which in turn forms the basis for stability. This is because a rule that has very low confidence may occur quite often, but its result will not be right. Confidence measures the reliability of the inference of a rule, hence the higher the value for a rule, the stronger the rule will be and the better the result that would be obtained. Table 4.6a showed the confidence for the first set of experiments when the number of nodes in the MapReduce was set to be constant.

From the data depicted in Table 5a, the confidence of the rules generated tends to decrease from 80% to 20% as the volume (number of records in this case) of the data increases in the first set of experiments. Also, the number of rules increases from 1568 to 21690, resulting in more complex rules which occur more frequently as the value for the support increases from 35 to 70. This changes in the value of the confidence depict that the model was unstable as the data was changing. In the second set of the experiment where the number of nodes in the MapReduce was varied, there was slight increment (from 83.2% to 83.9%) in the value of the confidence for the rules generated as the volume of the data increases. The number of rules generated reduces from 1068 to 216, resulting in more viable rules that occur more frequently as the value for the support increases from 39 to 80 (see Table 5b). This changes in the value of the confidence are minimal, and it depicts that the model was stable as the data was changing with an increasing number of nodes.

Table 5a: Support and Confidence at Constant Number of Nodes

No of Records	Support	Confidence (%)	Number of rules
71543.00	35	80	1568
115427.00	45	50	3820
279950.00	52	40	11322
396428.00	70	20	21690

Table 5b: Support and Confidence at Varying Number of Nodes

No of Records	Support	Confidence (%)	Number of rules
71543.00	39	83.2	1068
115427.00	59	83.6	856
279950.00	63	83.7	522
396428.00	80	83.9	216

This implies that the model was volatile when the volume of data was increased without a corresponding increase in the number of nodes in the MapReduce processing platform. The model was more stable in the second case, as there was no significant change in the value of confidence. Also, as the volume of the data increases, the pressure on the constant number of nodes of the MapReduce increases and this was the reason for the high value for the number of complex rules and their respective supports values. This pressure was drastically reduced in the second set of experiments, hence the insignificant increase in confidence values. This is because as the volume increases, the number of nodes also increases, and there is no much pressure on the nodes.

5. DISCUSSION

The existing system which is operated manually is characterized by unnecessary delays which have negatively affected the scheme. This manual processing is unfit for handling the recent upsurge in data which technological improvement is generating at high rates and volume. For instance, it takes three to twelve months to complete both registration and update of enrollees, it takes a similar period for the payment of capitation, co-payments, *per diem* and reimbursement of claims. Also, referral of patients that should take at most twenty-four hours takes days before being attended to. These and many more issues generated by corruption have beclouded the scheme, and the lack of appropriate data analytics tools is at the heart or centre of this problem.

Big data analytics of NHIS data was achieved by parallelizing the Apriori ARM algorithm using the MapReduce distributed data processing framework. ARM was chosen for this research because of its power to process qualitative data more efficiently, unlike the other data mining techniques. Although, ARM, when implemented in the orthodox way to process big data, has issues of generating complex rules which slow down the system. To this end, by parallelizing ARM to run in a distributed environment curbed these issues, and this was achieved using the MapReduce framework. This framework is the most successful computing platform for analyzing big data because it adopts a data centric approach of distributed computing with the thought of moving computation/processing closer to data. With this, a complex piece of data could be partitioned and loaded into different nodes that would process it at a faster rate. The results of this research were presented and evaluated in Section 4.

From the evaluation presented in Section 4, it was evident that the response time and throughput of the algorithm vary as the volume, velocity and variety of the data also varies. The summary of this result is shown in Table 6. This variation affects the entire analytics process (*i.e.,* the extraction, pre-processing, analysis and visualization). This evaluation was done in two phases: using a single node in the MapReduce framework during rules generation and varying the number of nodes in the MapReduce framework during rules generation. In the first instance, the throughput indicates that the system processes an average of 2796.53 records every second when the number of nodes in the MapReduce is constant. The average of the throughput was 8865.29 records per second when the nodes were increased in the second sets of experiments. Going by our experiments, the result implies that as the number of nodes increases, the time taken to process the records reduces drastically, and the throughput increased in return. From Table 4 and Fig 7 the throughput increases from 1482.89 records to 4246.87 for the first sets of experiments and increases from 4475.44 to 13880.81 in the second sets of experiments as the volume of data increases from 71543 to 396428. This increment occurs regardless of the variety and velocity of the data.

Table 6: Summary of the Result of this Research

	CE	CC	CR	CU	CA	VE	VC	VR	VU	VA
Response Time	0.86	0.86	0.88	0.88	0.87	0.441	0.093	0.224	0.247	0.251
Throughput	2364.71	1086.73	3834.52	3900.17	2796.53	6710.68	8473.54	9707.81	10569.13	8865.29

For the response time, there was also a significant difference in the two sets of experiments, and the average value was conspicuously different. The response time also increases in response to an increase in the volume of data, and the number of requests also increases in the first set of experiments. But decreases as the volume of data and the number of requests increases. In Tables 3 and Fig 6, the average response time for the first instance is 0.87seconds. In contrast, it is 0.25seconds in the second case irrespective of the volume, variety and velocity of the data. From the above, the performance of the system was evaluated using response time and throughput as metrics. In either case, as presented by the two sets of experiments above, the performance is good, and this implies that the delays that affect the processing of data submitted by the providers and enrollees to the HMOs, has been drastically reduced considering the performance metrics and the values gotten from the evaluation. This would ease the processing of the data and as a result improve the flow of resources (*i.e.,* steady flow of resources to the facilities) in the form of capitation, fee-for-service, *per diem*, co-payments, enrollment registers, updates and referrals. Thus, a steady improvement in the structures, processes and outcomes leading to improvement in the quality of services rendered to beneficiaries by the facilities.

6. CONCLUSION

In conclusion, this research developed a distributed algorithm for the processing of big data to enhanced decision making in organizations. This provided a parallel, distributed and non-invasive technique for large-scale processing of data. With this, data scientists would have a tool for processing both structured and unstructured data with high volume and velocity. It was tested with NHIS data in Nigeria to address the delay in the existing manual data processing system. The results show quick response time and high throughput. Thus, the processing time of NHIS data was reduced. This will enhance the flow of resources among stakeholders in the scheme, and thus, a steady improvement in the structures, processes and outcomes leading to improvement in the quality of services rendered to beneficiaries by the facility would be attained. In summary, the research addressed two important issues in data analytics:

1. ability to effectively derived meaningful insights from structured, semi-structured and unstructured data for effective decision making by stakeholders.
2. ability to drastically reduce delay in data analytics due to orthodox data processing techniques.

7. REFERENCES

[1] Mazumder, S., Bhadoria, R. S. and Deka, G. C. 2017. Distributed Computings in Big Data Analytics: Concepts, Technologies and Applications. Springer International Publishing, Gewerbestrasse 11, 6330 Cham, Switzerland.

[2] Yange, S. T., Soriyan, H. A. and Oluoha, O. 2017. Design of a Data Analytics Model for National Health Insurance Scheme. Journal of Health Informatics Africa, 4(1): 42-50.

[3] Mirkin, B. 2010. Core Concepts in Data Analysis: Summarization, Correlation, Visualization. Department of Computer Science and Information Systems, Birkbeck, University of London, Malet Street, London WC1E 7HX UK.

[4] Mouthaan, N. 2012. Effects of Big Data Analytics on Organizations' Value Creation. Unpublished MSc. A thesis submitted to the Department of Business Information Systems, University of Amsterdam.

[5] Zhang, H., Chen, G., Ooi, B.C., Tan, K.L. and Zhang, M. 2015. In-Memory Big Data Management and Processing: A survey. IEEE Transactions on Knowledge and Data Engineering, 27: 1920–1948.

[6] Famutimi, R.F. 2018. Design and Implementation of In-Memory Technique for Managing Big Data Complexities. An Unpublished Ph.D. Thesis Submitted to the Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria.

[7] Yange, S. T., Soriyan, H. A. and Oluoha, O. 2019. A Fraud Detection System for Health Insurance in Nigeria. Journal of Health Informatics Africa, 6(2): 64-73.

[8] Das, S., Sismanis, Y., Beyer, K.S., Gemulla, R., Haas, P.J. and McPherson, J. 2010. Ricardo: Integrating R and Hadoop. SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA, 987-998.

[9] Saxena, P. and Govil, K. 2013. An Effective Reliability Efficient Algorithm for Enhancing the Overall Performance of Distributed Computing System. International Journal of Computer Applications, 82(5): 30-34.

[10] Chen, C.L.P. and Zhang, C.Y. 2014. Data-Intensive Applications, Challenges, Techniques and Technologies: A survey on Big Data. Information Sciences, 275: 314–347.

[11] Gilbert, S. and Lynch, N. 2002. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. SIGACT News, 33: 51–59.

[12] Kumar, U. and Kumar, J. 2014. A Comprehensive Review of Straggler Handling Algorithms for MapReduce Framework. International Journal of Grid Distribution Computing,7 (4): 139-148.

[13] Davenport, T. 2014. Big Data at Work: Dispelling the Myths, Uncovering the Opportunities. Harvard Business Review Press, Boston, Massachusetts, USA.

[14] Gandomi, A. and Haider, M. 2015. Beyond the Hype: Big Data Concepts, Methods and Analytics. International Journal of Information Management, 35: 137–144.

[15] Etikala, P.R. 2016. Designing & Implementing a Java Web Application to Interact with Data Stored in a Distributed File System. An Unpublished M.Sc. Thesis Submitted to Graduate Faculty of St. Cloud State University.

[16] Sajwan, V. and Yadav, V. 2015. MapReduce: Architecture and Internals. International Journal of Science and Research, 4(5): 774-777.

[17] Kotsiantis, S. and Kanellopoulos, D. 2006. Association Rules Mining: A Recent Overview. International Transactions on Computer Science and Engineering, 32 (1): 71-78.

[18] Babi, C., Rao, M.V. and Rao, V. V. 2017. Study of Association Rule Mining for Discovery of Frequent Item Sets on Big Data Sets. International Journal of Applied Engineering Research, 12(22): 12169-12175.

[19] Agrawal, R. Imielinski, T. and Swami, A. 1993. Mining Association Rules Between Sets of Items in Large Databases. In the Proceedings of ACMSIGMOD Conf. on Management of Data, May 1993, 207-216.

[20] Aggarwal, C. and Yu, P. 1998. Online Generation of Association Rules. In the Proceedings of the 14th Intl. Conf. on Data Engineering, 402-411.

[21] Agrawal, R. 1994. Fast Algorithms for Mining Association Rules in Large Databases. Computer Science and Technology, 15: 487-499.

[22] Singh, S., Garg, R. and Mishra, P. K. 2014. Review of Apriori Based Algorithms on MapReduce Framework. In the Proceedings of 2014 International Conference on Communication and Computing (ICC - 2014), Bangalore, India, 593–604.

[23] Gautam, J. and Srivastava, N. 2015. Analysis of Medical Domain Using CMARM: Confabulation Mapreduce Association Rule Mining Algorithm for Frequent and Rare Itemsets. International Journal of Advanced Computer Science and Applications, 6(11): 224-228.

[24] Saabith, S. A., Sundararajan, E. and Bakar, A. A. 2016. Parallel Implementation of Apriori Algorithms on the Hadoop-MapReduce Platform- An Evaluation of Literature. Journal of Theoretical and Applied Information Technology, 85(3): 321-351.

[25] Oweis, N. E., Fouad, M. M., Oweis, S. R., Owais, S. S. and Snasel, V. 2016. A Novel MapReduce Lift Association Rule Mining Algorithm (MRLAR) for Big Data. International Journal of Advanced Computer Science and Applications, 7(3): 151-157.

[26] Singh, B. and Miri, R. 2016. An Efficient Parallel Association Rule Mining Algorithm based on MapReduce Framework. International Journal of Engineering Research & Technology, 5(6): 236-240.

[27] Ramteke, S. 2016. Association Rule Mining Algorithm Using Big Data Analysis. International Journal on Recent and Innovation Trends in Computing and Communication, 4(5): 73-75

[28] Nancy, J. J., Rani, M. J. and Devaraj, D. 2016. Association Rule Mining in Big Data using MapReduce Approach in Hadoop. GRD Journals|Global Research and Development Journal for Engineering/International Conference on Innovations in Engineering and Technology, 179-186.

[29] Prajapati, D. J., Garg, S. and Chauhan, N. C. 2017. Interesting Association Rule Mining with Consistent and Inconsistent Rule Detection from Big Sales Data in Distributed Environment. Future Computing and Informatics Journal, 2: 19-30.

[30] Bagde, P.R. and Chaudhari, M.S. 2016. Analysis of Fraud Detection Mechanism in Health Insurance Using Statistical Data Mining Techniques. International Journal of Computer Science and Information Technologies, 7 (2): 925-927.

[31] Devi, M. R. and Sarojini, A. B. (2012). Applications of Association Rule Mining in Different Databases. Journal of Global Research in Computer Science, 3(8): 30-34.

[32] Deshpande, D. & Anami, B. S. 2016. Association Rules Based Analysis for Arthritis Patients' Data. International Journal of Modern Trends in Engineering and Research, 3(3): 388-393.

[33] Kang'ethe, S. M. and Wagacha, P. W. 2014. Extracting Diagnosis Patterns in Electronic Medical Records using Association Rule Mining. International Journal of Computer Applications, 108(15): 19-26.

[34] Al-Maolegi, B. A. 2013. An Improved Apriori Algorithm for Association Rules. International Research Journal of Computer Science and Application, 1: 1-8.

[35] Rao, P. G. 2012. Implementing Improved Algorithm Over Apriori Data Mining Association Rules Algorithm. International Journal of Computer Science and Technology, 3: 489-493.

[36] Woo, J. 2012. Apriori MapReduce Algorithm. International Conference on Parallel and Distributed Processing Techniques and Applications, 20-31.

[37] Mahendra, N. T. 2012. Data Mining for High Performance Data Cloud using Association Rule Mining. IOSR Journal of Computer Engineering, 16-22.

[38] Hegazy, O. Y. O. 2012. Ann Efficient Implementation of Apriori Algorithm Based on Hadoop MapReduce Model. International Journal of Reviews in Computing, 12: 59-67.

[39] Dean, S. G. 2008. MapReduce: Simplified Data Processing on Large Clusters. ACM, 5: 107-113.

[40] Leem, H. K. (2012). Parallel Data Processing with MapReduce: A Survey. ACM, 40: 11-20.

[41] Yang, A. H. 2007. MapReduce Merge: A Simplified Relational Data Processing on Large Clusters. ACM, 1029-1040.

[42] Maitrey, S. and Jha, C. K. 2015. MapReduce: Simplified Data Analysis of Big Data. Procedia Computer Science, 57: 563 – 571.

[43] Kumar, L., Rajawat, S. and Joshi, K. 2015. Comparative analysis of NoSQL (MongoDB) with MySQL Database. International Journal of Modern Trends in Engineering and Research, 2(5): 120-128.

[44] Pothuganti, A. 2015. Big Data Analytics: Hadoop-Map Reduce & NoSQL Databases. International Journal of Computer Science and Information Technologies, 6 (1): 522-527.

[45] Yange, S. T., Soriyan, H. A. and Oluoha, O. 2019. An Implementation of a Repository for Healthcare Insurance Using MongoDB. Proceeding of the 14th International Conference of Nigeria Computer Society (NCS), Gombe, Nigeria, 30: 54-67.

[46] Ananth, G. S. and Raghuveer, K. 2017. A Novel Approach of Using MongoDB for Big Data Analytics. International Journal of Innovative Studies in Sciences and Engineering Technology, 3(8): 7-12.

[47] Yange, S. T., Soriyan, H. A. and Oluoha, O. 2017. A Schematic View of the Application of Big Data Analytics in Healthcare Crime Investigation. Journal of Health Informatics Africa, 4(1): 32-41.

[48] Oyegoke, T.O. 2015. Development of an Integrated Health Management System for National Health Insurance Scheme. An Unpublished M.Sc. Thesis Submitted to the Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria.

[49] Eteng, F.O. & Ijim-Agbor, U. 2016. Understanding the Challenges and Prospects of Administering the National Health Insurance Scheme in Nigeria. International Journal of Humanities and Social Science Research, 2(8): 43-48.

[50] Alimi, O. M., Binuyo, O. G., Gambo I. G. & Jimoh, K. 2016. Realtime National Health Insurance Scheme (RNHIS): Means to Achieve Health for All. International Journal of Computer Science, Engineering and Applications (IJCSEA), 6(2): 1-8.

[51] Oyegoke, T. O., Ikono, R. N. and Soriyan, H. A. 2017. An Integrated Health Management System for National Health Insurance Scheme in Nigeria. Journal of Emerging Trends in Computing and Information Sciences, 8(1): 30-40.

[52] NHIS (National Health Insurance Scheme), 2013. National Health Insurance Scheme Operational Guidelines. Accessed on 01.06.2017 from http://www.nhis.gov.ng/images/stories/hmoregister/NHIS_OPERATIONAL_GUIDELINES.pdf.

[53] Hoyt, R.E. and Yoshihashi, A. (2014). Health Informatics: Practical Guide for Healthcare and Information Technology Professionals, Sixth Edition. Pensacola, FL, Lulu.com.

[54] Ebenezer J. G. A. and Durga, S. (2015). Big Data Analytics in Healthcare: A Survey. ARPN Journal of Engineering and Applied Sciences, 10(8): 3645-3650.

[55] Olaniyan, A. O. 2017. Assessment of the Implementation of National Health Insurance Scheme (NHIS) in South-Western Nigeria. Unpublished PhD Thesis submitted to the Department of Public Administration, Obafemi Awolowo University, Ile-Ife, Nigeria.