IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Handling Big Data using Data-Aware HDFS and Evolutionary Clustering Technique

[1]Kusuma K P, [2]Swomyashree K M

[1]Student, [2]PROFESSOR

[1] Department of MCA

[1] PES College of Engineering, Mandya, Karnataka, India

Abstract: The increasing usage of digitally enabled frames and the Internet of Things has resulted in a wealth of knowledge and a diversity of structures. The environment or document frameworks presented by Hadoop are utilised in most enormous data arrangements (HDFS). In any case, the reflectors showed weaknesses in such frameworks in the maintenance of current data. While some research has resolved these concerns for explicit diagram data variants, existing data are based on reasonable data. Studies show that productivity concerns lead to significant scope problems, such as increased space in server farms and waste in assets (such as the use of force), leading to ecological problems (such as more fuel byproduct) [1]. We propose a Hadoop Eco-Focus module that is information-rich. Also, we propose a genetic algorithm encoding strategy that uses circulating encoding. Our architecture enables Hadoop to manage data distribution and organization, as well as conduct group analysis on subsets of data. Even when improvement time and use of assets are involved, we can manage a wide range of data types. We have tested our assumptions on a number of LUBM datasets.

Index Terms- Strategies for clustering, Distributed computing, Information Management and Management, Optimization and Scalability

I. INTRODUCTION

There are different challenges in creating science from knowledge. One of the main challenges is that these data are large, dynamic and varied and come from a variety of different sources and often have a standard design.

Most current data tests, board devices and administrations were developed for usage as an information reserve using the Hadoop Distributed Filing System (HDFS) and now use Hadoop eco-framework administrations for preparation. These insightful devices. Hadoop performs in terms of value and performance admirably.

According to[1] Hadoop's adaptability makes it easier to manage the information, which is why customers are inefficient. The method by which clients add machines to solve computation problems caused them to pay less attention to how their routines use assets, and (b) a large number of HDFS clients believe it is designed for clusters that are cussing, according to Huang et al. (a). It is now acceptable to leave the scripts in the background for a longer time without looking at the resources used by these cycles.

The inventors of Bajda-Pawlikowski etalHad.'s apt [2] showed such a failure and denied it by a factor of fifty for prescribed information. According to his blog [3], however, project information lightning is predominantly semi-structured, multi-structured and unstructured. According to the IDC, the digital data volume will increase annually between 40 and 50% [4]. According to IDC, the total is expected to be 40 zettabytes by 2020[4] (ZB). By In 2020, several times the volume and the quantity of knowledge containers will have been produced in the world [3]. It is necessary to expand and process the present data analysis tools to Big Data to maximize the use of resources.

Rohloff et al. [5] explained the process in 2011 by using triple representations to store graphic data in Hadoop. They showed how to adapt sub graph patterns very quickly on knowledge graphs. Due to the paper's major focus, the ideas offered can be adapted to various types of graphs. From this paper came the SHARD system. Because of its methods, Hadoop can scale the pattern for sub-graphs. Huang et al works were published in 2011 about the Scalable SPARQL query of huge RDF graphs. Showed that the Rohloff et al. [5] techniques had a difficulty with efficiency. Rohloff et al.[5] found Huang et al.[1] to be 1340 times less effective in producing a pattern matching element in a Hadoop-based System than other alternative strategies.

When it comes to storing Big Data, some solutions do not use HDFS at all their horizontal scaling mechanism is similar. There are alternatives that work with HDFS's core and are potentially generalizable in those situations that we presented and tested. Apache Spark [6] and Mesos [7] are two examples of technologies that use HDFS for storage. HDFS and Yarn resource negotiator are used in the HAMR [8] system to support Hadoop. With the proposed HDFS data consciousness architecture for a wide number of current Big Data applications, this would boost HDFS optimization.

Introducing Spark [6] and Storm [9]. Another large-data solution, Apache Storm [9], uses yarn to conduct an analysis in real time on unlimited data streams. Storm builds on the work of Hadoop in the execution of instructions. Storm is somehow optimized. [10] Is an example of optimizing schedules, while [11] proposes extensions to Storm. The objective of this research is to optimize

HDFS as a warehouse for information However, Storm[9] analyses knowledge streams in other circumstances and saves results in HDFS for further processing, in which our method makes Hadoop data analysis tools easier to use.

Our main objectives was to better Hadoop's distributed grading system (HDFS) efficiency for the processing of current data and optimize HW resource utilization for semi/multi/unstructured growth and efficiency. While there are generalized SHARD [5] and Scalable SPARQL [1], considerable limitations exist in how the modern data are handled. In addition, the constant flux of dynamic changes in the data has its limits. Hajeer et al.[12] cite scalability and storage as examples of these constraints.

Processing the data into vertex-to-vertex triple data points, and then adding cluster affiliation data to these triples in order to produce quadruples, as specified in the architectural section. These strategies have enabled (1) the collection and transformation of quad data from a number of sources, (2) dynamically transforming data and converting it into databases of graphs and (3) the preparation of data for use in a new version of the Hajeeret al. [12]. A single chromosome encoding in conjunction with a novel crossover, mutation and assessment procedures has been used to address the current data clustering challenge to meet the requirements of the new distributed encoding technique. We then dispersed the sub graphs over HDFS and supported cluster affiliations in order to provide optimal questions and processing data.

Subject	Predicate	Object
Alice	knows	Mike
Alice	Likes	Photo1
Mike	Likes	Photo1
Bob	Likes	Photo1
Bob	Knows	John
Bob	Interested In	Topic

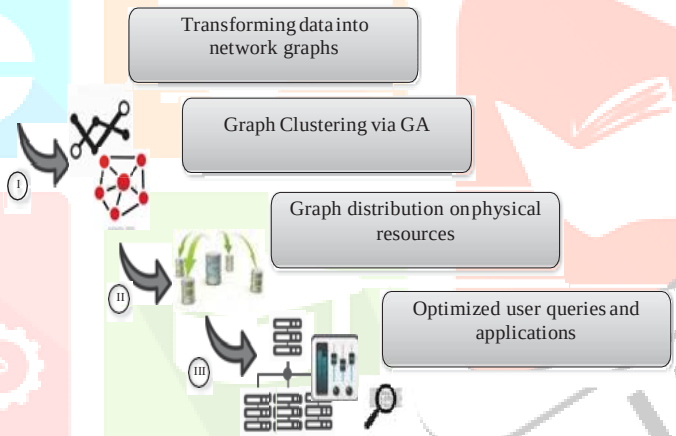

Fig.1. Computational steps of the proposed framework.

The contribution and modules for the framework suggested are shown in Fig. 1: (2) The module will distribute the information into the appropriate data blocks after finding patterns in the grapes, (3) distribute the blocks to the corresponding machine, and (4) the optimized DHFS will be a source of knowledge for querying services and providing the graphing platform (1)

In brief, the suggested solution indicates that HDFS flexibility with regard to the administration of current data, combining data sensitivity modules that recognize, distribute and manage data across a scalable categorizing system. This framework can optimize the Hadoop eco-system and employ extra tools and services using HDFS as distributed storage more efficiently.

New investigations have provided exciting solutions in the future generation of analytics and lambda architecture. Song et al.[13] examined recent research on data kinds, storage models, analytical approaches and Internet applications. They also discussed issues and developments in the analysis of massive amounts of data in order to forecast current and future trends. Song et al.[13] demonstrated how streaming and real-time information has risen with the growth in internet streaming and how a SQL based DBStream [13] system is continually analyzing data using surveys.

II. SCOPE OF OBJECT

Present for inefficient systems require more space in data centers as well as certain serious environmental effects as a result of increased emissions of carbon as a result of increased energy consumption [1]. This will affect companies because the electricity consumption and performance of the same hardware resources are increased. We want to scale systems efficiently.

The theory of graphs could be an area highly investigated. The graph theory is a critical method of understanding and using large numbers of data, according to Dr. Roy Marsten's blog [23]. The theory of graphs could be an area highly investigated. The graph theory is a critical method of understanding and using large numbers of data, according to Dr. Roy Marsten's blog [23].

Consequently, "Google created an online programmer that outperformed its current competitors dramatically, causing 'Google' to become a verb"[23]. Graphs can be created using a wide range of data. But on the other hand, a lot of problems will be turned into graphic problems. Most of these difficulties are resolved rapidly through graph theories and algorithms. We See SHARD as an advantage for current data in Rohloff et al [5], as we suggest how to turn multiple types of data into different graphs as illustrated in the Design section

As described in the graph databases section, graph data scale techniques using distributed eco-systems such as Hadoop are available. Moreover, Huang et al. [1] succeeded even better by adapting Rohloff et al. [5]'s scholarship to include RDF data and to overcome Rohloff ET limitations in terms of methodology.

Previous efforts were made towards optimizing graphic findings, for example the hash-division of SHARD data [5]. Hashes on the other hand are limited to objects in RDF graphs, because intermediate information must be sent across the network. Objects connected with a theme during one or two hops between subject and object were turned into similar blocks in a study by Huang et al. [1]. (one to 2 edges travel distance between subject and objects). Space limits were, however, related to the growth in data size. In addition, a densely connected diagram has many disadvantages to such an approach. Other projects such as Sempala [24], HIVE, PigSPARQL [25] & [26], Map Merge [27], and MAP-SIN [28] were overcome to some extent by scalability. Nevertheless, these works benefit from MR, HIVE and Impala, where our frameworks improve partitioning and have a unique storage system more than three times.

RDF graphs are stored by Sempala, PigSPARQL, Map Merge and MAPSIN in a number of ways. These frames convert each predicate into columns, create tables with appropriate patterns and eventually convert three times to regular database records. These frameworks limit the ability to update all data when new predicates, data and update schemes are available. This makes it easier to distinguish between RDF and the concept of graphical databases where updates contain new predicates.

2.1 Graph Databases

Traditional databases have issues storing and executing current and emerging applications, notably the relational format. Graphic databases have become more popular and a subject that was almost neglected in the early 1990's [29] reappeared. The relevance of such databases evolved because information in modern data is based on relationships that are sometimes equally or perhaps based on data from entities [30]. These databases have been the focus of numerous industries' efforts (e.g. Biology [31], semantic web [32], web mining [33] and chemistry [34]).

According to Silberschatz et al.[35], the most fundamental definition of an information model is a series of conceptual instruments used to shape the real world and its relations. From the database standpoint, three models are: (1) collection of data structures; (2) set of rules of integrity; and, (3) operator and interface rules [35].

An RDF store is a graphical database, which stores and retrieves triples via semantic zed queries. A triple is a subject, predicate and object data entity. Triple stores data as triple and finds it through a question language. However, they differ in several ways from relation databases, the most important one being that a triple store is optimized for three times.

Figure 2 shows in a triple store three times. There has been some progress on clustered RDF database systems. The SHARD [5], YARS2[36], Jena and Jena Elephas[37] and Virtuoso[38] clustered RDF databases have a hash partition that can be tripled through several nodes of the computer and parallel access to them at query time.

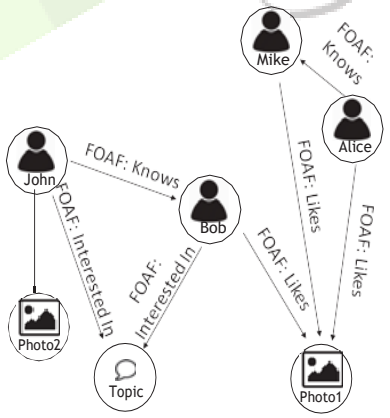

Fig.2 RDF Triple store and its representation as a graph

2.2 Community Detection and Multi-Objective Evolutionary Algorithms

Graph theory categories often develop algorithmically when particular metrics of density and scarcity are optimized by an algorithm, which divide the network into communities [39]. Optimization of these measures is NP-hard in many instances. Approximations of NP-hard situation but faster technology are generally applied. Evolutionary algorithms (EA) as defined in [40] are one of the sensible methods of NP-hard resolution. Genetic algorithms (GAs) have been used to detect communities in a number of publications [41], [42], [43], [44], [45], and [46], and EAs have been used in a number of publications [47] and [48]. Chang et al. [49] describe how ant colony optimization can be used to detect communities of insects.

The Girvan-Newman algorithm [56] is a popular community-founding algorithm that removes edges with the highest centrality from the network, until the edges remain. For example, if nc is the total number of communities, then equation (1) [57] will define modularity as follows: This is m edges in a graph and li edges in a community. Do I equal the sum of all node degrees?

$$Q = \sum_{i=1}^{n_c} \left[\frac{l_i}{m} - \left(\frac{d_i}{2m} \right)^2 \right]$$

"Genetic algorithms (GA) develop into better answers for the population of chromosomes that encoding prospective solutions/individuals to an optimization issue" says Yi et al.[58]. After the response in the chromosome format is genetically represented, GA then randomly / definitely initiates a solution population and determines the fitness functions. Then, through repeated applications of numerous genetic operators such as selection, crossover, and mutation, GA attempts to improve it. Finally, the results are optimized with the help of local search and border search operators."

2.3 Service deployments over HDFS

As previously indicated, HDFS is a data source that distributes contemporary Big Data solutions such as Apache Spark [6], Meso [7] or HAMR [8]. In many deployments of such solutions, HDFS or a service that utilizes HDFS is used.

III. A PROBLEM WITH HDFS'S PERFORMANCE AND EFFICIENCE

The Hadoop eco-ability system for the management of company data and construction apps is established by case and hence by data. Since IT BI teams are configuring those systems to meet their objectives and road plans in firms, they focus on information and use cases. They focus on information and cases.

Most business data are collected for specific goals. These data are then stored on storage devices, waiting to be used by the BI team, which leads to data from a number of sources with multiple structures. In the case of company uses and therefore data, the Hadoop eco-ability management system for company data and creating apps is determined. As IT BI teams in corporations and companies construct these systems so that they may meet their aims and roadmaps, they concentrate on information and use cases.

According to Huang et al. [1] and Rohloff et al. [5, there is no graph optimization in Hadoop. HDFS's inefficiency can be attributed to a number of factors, including: First, due to the fact that a set of computing resources is geographically isolated from each other due to Hadoop's default hash partitioning, completing graph operations will require a significant amount of data transfer. As a consequence, the grouping of similar data can be a victory, according to [1]; (2) Hadoop considers that all data blocks and partitions are of equal value so that the closeness of the cluster neighbors, and Physical closeness, efficiency increases, and (3) graphic data is not optimized by HDFS.

In a Hadoop-based system, the Rohloff et al. technique [5] presented Huang et al. [1] an efficacy problem. According to Huang et al. [1] as well as Rohloff et al. however, have taken a generalized approach to solving the problem. We believe that the technique has some weaknesses in addressing vast and dynamic un/semi/multi-structured data because it concentrates on one type of file and one clustering algorithm. These limitations were validated in a recent study by Hajeer et al.[12]. Using genetic algorithms, it was shown how to cluster data in this way. In Hajeer et al. [12], Pizzeria [43] and [42], as well as an inventory of other works, such as [48], [46], [62], and [63], we applied the metamorphosis method to turn desirable data into graph data. [12] Huang and colleagues [1] and Rohloff and colleagues.

IV. INDICATORS FOR HDFS THAT ARE AWARE OF DATA

4.1 Graph Transformation

An overview of modern data sources and problems was provided in the introduction to the book. It has been made clear that data might come from a variety of sources. Assume that SN stands for source SN and ZS1 for ZS1 in order to calculate S. Each of these diverse sources generates or contains data with a distinct structure, for the same entity sometimes, but with different structures and data.

Where DSZ is the structure of knowledge, D= (DS1, DS2,...,DSZ). In the infinite superset DS, |DS|=derived from Z. DSZ (Structure) Since G (V, E) is an undirected diagram with m vertices and n borders, D was transformed into Data. More information about this change can be found in the section titled "Overall Architecture."

4.2 Graph Clustering

We referred to a diagram with the V and E G vertices (V, E). Also clustering C= (C1, C2, C3,..., CJ) is based on The number of vertices!=m and edges |E |=n, as V partition, as set by disjoint. C is a G cluster composed of the J clusters. The j number of Clusters ranges between 1 and m, with j=1 that indicates that C has only one C1 subset = V and j=m that only one vertex is contained in each CJ cluster. Cluster Cj is referred to as the G.G.[Cj]:=(Cj, E(Cj) subsection in which CJ represents the edges of the intracluster, and E/E(C) represents the edges of the intracluster. The number of borders within the cluster is denoted by the m(C) and N represents the number of borders between the clusters (C).

In Hajeer et al. [12], we employed modularity as a fit-ness measure in our clustering approach. For random graphs with the same node degree distribution as the predicted number of borders inside groups 1 and 2 present network, minus the fraction of edges that fall within group 1 or 2, is then used to calculate modularity Q. As a result, Avw-(kV kW)/2m is the particular Number of edges from v to w minus the predicted number of edges between them. Equation (2) [57] can be used to express modularity.

4.3 Graph Distribution and Assumptions

Three key challenges were experienced in optimizing HDFS, two of them how Hadoop hash and dispense data. Our hypothesis and research showed (1) intracluster data storage on one single machine and (2) close intercluster data storage on nearby machines is an important step towards improved HDFS. Let M=(M1, M2,....,MI) be a group that belongs to a number of natural finite numbers. And Mi, Mi+1, Mi+n are machines where the distance between the physical network and Mi+1 is lower than Mi

and Mi+n. The Cj cluster should be [0, of computers representing Hadoop computing resources, where I am in a single graph partition or at least on the same Mi. When Mi is out of place, it should be moved to Mi+1 and on; the closer the machine is, the more priority. results. When Cj and Cj+1 have more inter-cluster rims than Cj and Cj+n, the m, Mi and Mi+m (physically closest), where 0mI, should be positioned as small as possible.

V. EXPERIMENTS AND RESULTS

The work is separated into two main areas: firstly, the graph shop algorithms were developed and tested to verify the outcomes of the clustering; and second, testing and comparison of the impact of the optimizing framework on HDFS. In Table [66], All trend models and graphs are created.

5.1 Graph Conversion and Clustering

We checked that our approach to clustering was correct and that the results were valid and comparable. We have carefully selected some well known smaller data sets and twice checked for the same data sets used in previous comparison studies. These are the following sets:

Zachary Karate Club: This graph contains 34 vertices and 78 edges. University karate club members are represented by nodes and communication patterns are represented by connections. It was collected definitely in 1997[67].

Bottlenose Dolphins: In 1994, a network of 62 dolphins and their interactions was established using seven-year-long data. U.S. Political books: 105 nodes network with 441 boundaries. The Network refers to a collection of frequently purchased books on US policy.

American College football: The teams are represented by 115 nodes and they are connected by 613 edges.

TABLE 1 shows validation results and compares them with some of the most popular algorithms. In some cases our technique has been as modular as possible, whereas in others it has been as modular as possible. They were excluded due to the great modularity of some methods; the results are as follows:

Dataset	GN	CNM	L Max	GATHB	MOG A-Net	**Our Method**
Karate	0.4	0.380	0.419	0.4	0.416	**0.416**
Dolphins	0.52	0.495	0.523	0.52	0.505	**0.528**
Football	0.6	0.577	0.61	0.55	0.515	**0.539**
Books	0.51	0.502	0.526	0.52	0.518	**0.523**

The TABLE 1 and [68] results show that our approach converges on ideal solutions and therefore the quality of the results is generally better (compared to other popular algorithms). Some examples were less modular than others.

We have found that selection influences how the solutions converge. In addition, binary selection converges in certain datasets to a higher modularity in a relatively short time, during random selection; local optimum fitness can deliver a quicker rate for jumps.

The results of evaluations of each population were provided to examine the convergence of solutions over generations. By dumping the population array, we have been able to generate graphs and calculate trend models through the dispersion plot for each generation to generate a tangible representation of outcomes. This relationship between modularity distribution and the number of generations required to extract modularity has been established in our research.

For expanding our Big Data strategy, we used LUBM to generate RDF graph data, which we deployed in a cluster with the properties listed in TABLE 2. From the configurations we used, we had 87 containers. All or all 48 discs, as well as two CPU cores and four gigabytes of memory are allowed into each container.

We used six computer nodes instead of ten or twenty compared to previous studies. In order to assess whether the networking and distance impact within the network was real, we simply examined the number of nodes. However, it makes more sense with YARN and therefore the concept of containers to Comparing resources to the number of nodes. Our cluster and computing process configuration lead to 86 containers, each with 4 GB of memory and a couple of CPU (one core) threads (max 2.93 GHz and min of 2.00 GHz), totaling 344 GB of memory and 172 CPU threads (86 centres), as opposed to Huang et al80 GB of memory, 40 CPU and 40 discs equally distributed among 20 computer nodes.

TABLE 2 HADOOP CLUSTER AND CONFIGURATIONS

Machine		Threads	Memory	Disks
Master	Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz	72	64	10
Node1	Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz	56	64	10
Node2	Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz	40	64	10
Node3	Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz	40	64	10
Node4	Intel(R) Xeon(R) CPU X5570 @ 2.93GHz	16	96	2
Node5 CPU	Intel(R) Xeon(R) E5520 @ 2.27GHz	16	48	6

The LUBM may be university domain taxonomy for synthetic OWL and RDF data, with 14 queries representing a variety of properties and scalability on any scale. The most popular benchmark of the Semantic Web community is LUBM.
To evaluate the behavior of our system, we have built various data sets of varying sizes. Before clustering, we deleted valuable "type" or similar predicates that could have previously been used to improve cluster standard and reduce graph complexity [1]. We examined the time needed to initialize a population of solutions with 1000 population each generation. The time to carry out the graph conversion and thus to initialize the primary population is shown in TA- BLE 3. The cost of spending too much data preparation time can be offset by the amount of data processing and querying operations which new technologies [22]. It's important to remember that in the future, when data is heavily processed or queried, our framework is best used. It could be a good idea to set up a system for fast reaction and low overhead hardware.

TABLE 3 POPULATIONS INITIALIZATION & ALGORITHM RUN TIME (LUBM DATASETS)

Number of triples	Initialize Popula-tion (S)	Algorithm Run Time (Minutes)
8,970,048	13.556	21.7
20,637,270	19.621	31.6
30,285,222	28.611	47
221,140,408	207.314	261(~4.3 hours)

In the LUBM data we probed deeper. Despite the large number of triples, it doesn't take a long time to initialise the primary population with random inter-cluster membership. Fig. 3 demonstrates population convergence to maximum modularity over time for a set of 30M quads.

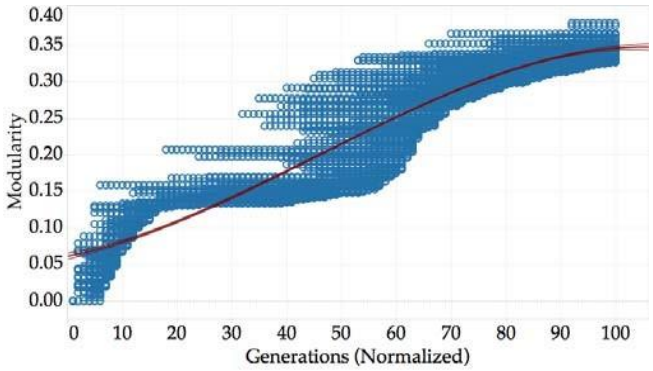

Fig 3. Convergence to maximum modularity (LUBM 30 million Triple)

The 30 million triple data trend model for LUBM is illustrated in Fig. 4

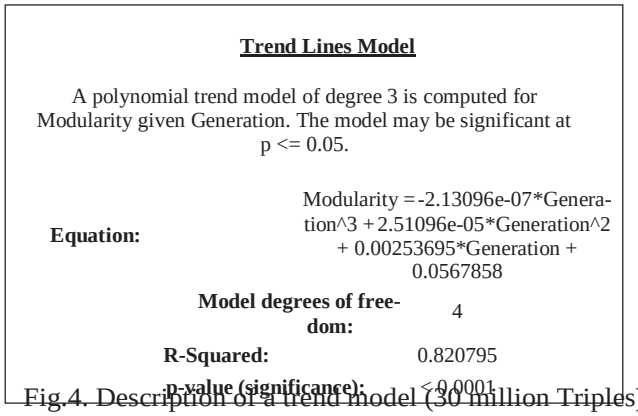

Fig.4. Description of a trend model (30 million Triples).

Using the encoding technology we provided, we have generated a solution area with numerous local solutions. By adopting a 100% mutation rate and multipoint crossovers, we were able to avoid falling into optimal local solutions. When the mutation rate is set to 100 percent, there is no variance in the modularity of a specific solution. On the other hand, the likelihood of answer is less than 100% modified thanks to the proposed encoding (The information used affected 72% of the solutions). Traditional encoding is influenced more by mutation than the encoding proposed. The next rate of mutation is therefore needed to influence solutions.
Figure 3 shows the modularity and the number of all generations (Each bin could have a range from bin x-value to bin x-value following). Since the majority of intra-cluster borders do not affect the number of communities generated, including solutions, have little impact on general health, which explains the high non-maximum modularity rates. However, The four-fold response did not influence the algorithm's ability to save from similar events.

5.2 Experiments in System Performance

In this section we analyse our system's performance after clustering and positioning against a variety of RDF stores, including SHARD. We have utilised Cloudier Impala to generate a data table. We concentrated on time and resources. When you request our frame to split our cluster, the system setup employed is the same as in TABLE.
We use LUBM to build a dataset in our experiments. The dataset created, with N-Triples sizes ranging from 37 to 142 GB and 200 to 600 million tonnes, demonstrates the complexity of each query by comparing Number of benchmark connections in every query.

VI. CONCLUSION

We introduced an HDFS data acknowledgement and, consequently, HDFS services in this post to optimize cutting-edge RDF storage. In order to manage the physical position of the information according to the graph location owing to the causality of HDFS processes, a cluster based data dividing was provided. This enables less resource-intensive HDFS data search multi-processing. Our system was ready to perform faster than certain attempts for scalable RDF data storage and slightly slower than others. It consumes less resource, however. Testing of lambda and next-generation analysis [15], [16], [17] and [19], as well as Apache Kudu[20], in conjunction with a series of investigations [25] have proven that the processing of OLAP workloads was quick and efficient and they performed well in performing time sensitive workloads. However, it is worthwhile reviewing the influence of intelligent data placement on such strategies. In order to reduce processing costs, we intend to improve the distributed encoding and genetic engineers in the future. Furthermore, we want to use the lambda and next generation analytical tools and frameworks introduced in recent studies to experiment with dynamic updates to increase data flow speeds. The causality in HDFS processes also matches the location of the graph. This enabled less resource-intensive multiprocessing of HDFS data searches. Our framework for scalable RDF data storage has been able to perform some attempts while being considerably slower than others. It consumes fewer resources, however. Research on lambda architecture[15] and next generation analytics[16],[17] and [18], as well as on Apache Kudu[20], and a number of tests[21] have been shown that OLAP's workloads might be faster and more efficient and time-critical workloads executed. Time and effort are well worth it. The influence of smart data placement on such systems should nevertheless be examined. In the future, we aim to improve distributed encoding, thus reducing genetic operators' overhead processing. We also use lambda architecture and tools and frames for next-generation analysis that we were able to test for faster information flow through dynamic updates.

REFERENCES

[1] J. Huang, D. J. Abadi and K. Ren, "Scalable SPARQL querying of enormous RDF graphs," Proceedings of the VLDB Endowment, vol. 4, no. 11, pp. 1123-113, 2011.

[2] K. Bajda-Pawlikowski, D. J. Abadi, A. Silberschatz and E. Paulson, "Efficient processing of information warehousing queries in an exceedingly split execution environment," in Proceedings of the 2011.

[3] M. Walker, "Data Science Central," 19 Dec 2012. [Online]. Available: http://www.datasciencecentral.com/profiles/blogs/structured-vs-unstructured-data-the-rise-of-data-anarchy. [Accessed 16 Oct 2015].

[4] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and largest growth within the Far East," IDC iView: IDC Analyze the long run, vol. 2007, no. 2012, pp. 1--16.

[5] K. Rohloff and R. E. Schantz, "Clause-iteration with Map Reduce to scalable query data graphs within the SHARD graph-store," in the fourth proceedings international workshop on Data- intensive distributed computing, 2011.

[6] T. A. S. Foundation, "Apache Spark," The Apache Software Foundation, [Online]. Available: http://spark.apache.org. [Accessed Jan 2016].

[7] T. A. S. Foundation, "Apache Mesos," The Apache Software Foundation. [Online]. Available: http://mesos.apache.org. [Accessed 19 Jan 2016].

[8] E. I. Inc., "HAMR - Faster than the speed of knowledge," ET International, Inc., [Online]. Available: http://www.hamrtech.com/index.html. [Accessed 19 Jan 2016].

[9] Apache Storm, "Apache STORM," Apache Software Foundation, [Online]. Available: http://storm.apache.org. [Accessed 16 9 2016].

[10] L. Aniello, R. Baldoni and L. Querzoni, "Adaptive online scheduling in storm," in Proceedings of the 7th ACM international conference on Distributed event-based systems, 2013. P. Basanta-Val, N. Fernandez-Garcia, A. Wellings and N. Audsley, "Improving the predictability of distributed stream processors," Future Generation Computer Systems, vol. 52, pp. 22--36, 2015.

[11] M. Hajeer, D. Dasgupta, A. Semenov and J. Veijalainen, "Distributed evolutionary approach to data clustering and modeling," in Computational Intelligence and data processing (CIDM), 2014 IEEE Symposium, 2014.

[12] H. Song, P. Basanta-Val, A. Steed, M. Jo and Z. Lv, "Next- generation big data analytics: State of the art, challenges, and future research topics," IEEE Transactions on Industrial Informatics, p. In Press, 2017.

[13] N. Agnihotri and A. K. Sharma, "Proposed algorithms for effective real time stream analysis in big data," in Image scientific discipline (ICIIP), 2015 Third International Conference on, 2015.

[14] L. Aniello, R. Baldoni and L. Querzoni, "Adaptive online scheduling in storm," in Proceedings of the 7th ACM international conference on Distributed event-based systems, 2013.

[15] P. Basanta-Val, N. Fernandez-Garcia, A. J. Wellings and N. C. Audsley, "Improving the predictability of distributed stream processors," Future Generation Computer Systems, vol. 52, pp. 22--36, 2015.

[16] P. Basanta-Val and M. Garcia-Valls, "A distributed real-time java-centric architecture for industrial systems," IEEE Transactions on Industrial Informatics, vol. 10, no. 1, pp. 27--34, 2014.

[17] P. Basanta-Val, N. C. Audsley, A. J. a. G. I. Wellings and N. Fernandez-Garcia, "Architecting Time-Critical Big-Data Systems," IEEE Transactions on Big Data, vol. 2, no. 4, pp. 310-- 324, 2016.

[18] M. Congosto, P. Basanta-Val and L. Sanchez-Fernandez, "T- Hoarder: A framework to process Twitter data streams," Journal of Network and Computer Applications, vol. 83, pp. 28--39, 2017.

[19] T. A. S. Foundation, "Introducing Apache Kudu," The Apache Software Foundation, 2017. [Online]. Available: https://kudu.apache.org/docs/. [Accessed 5 April 2017].

[20] N. Marz and J. Warren, Big Data: Principles and best practices of scalable real-time data systems, Manning Publications Co., 2015.

[21] M. Ferron-Jones, "It Peer Network," 16 May 2017. [Online]. Available: https://itpeernetwork.intel.com/new-breakthrough-persistent-memory-first-public-demo/. [Accessed 1 June 2017].

[22] E. Roy Marsten, "Is graph theory key to grasp Big Data," March 2014. [Online]. Available: http://www.wired.com/insights/2014/03/graph-theory-key- understanding-big-data/. [Accessed Oct 2015].

[23] A. Schatzle, M. Przyjaciel-Zablocki, A. Neu and G. Lausen, "Sempala: Interactive SPARQL query processing on hadoop," The Semantic Web--ISWC 2014, pp. 164--179, 2014.

[24] A. Schatzle, M. Przyjaciel-Zablocki, T. Hornung and G. Lausen, "PigSPARQL: a SPARQL query processing baseline for large data," in Proceedings of the 2013th International Conference on Posters Demonstrations Track-Volume 1035, 2013 A.Schatzle, M. Przyjaciel-Zablocki and G. Lausen, "PigSPARQL: Mapping SPARQL to Pig Latin," in Proceedings of the International Workshop on Semantic Web Information Management, 2011.

[25] M. Przyjaciel-Zablocki, A. Schatzle, E. Skaley, T. Hornung and G. Lausen, "Map-side merge joins for scalable SPARQL BGP processing," in Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on, 2013.

[26] A. Schatzle, M. Przyjaciel-Zablocki, C. Dorner, T. Hornung and G. Lausen, "Cascading map-side joins over HBase for scalable join processing," SSWS+ HPCSW, p. 59, 2012.

[27] R. Angles and C. Gutierrez, "Survey of graph database models,"ACM Computing Surveys (CSUR), vol. 40, no. 1, p. 1, 2008.

[28] R. Angles, "A comparison of current graph database models," in Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on, 2012.

[29] B. A. Eckman and P. G. Brown, "Graph data management for molecular and cell biology," IBM journal of research and development, vol. 50, no. 6, pp. 545-560, 2006.

[30] J. Hayes and C. Gutierrez, "Bipartite graphs as intermediate model for RDF," within the Semantic Web--ISWC 2004, Springer, 2004, pp. 47-61.

[31] A. Schenker, Graph-theoretic techniques for online page mining, World Scientific, 2005, p. 62.

[32] A. Nayak and that i. Stojmenovic, Handbook of applied algorithms: Solving scientific, engineering, and practical problems, John Wiley & Sons, 2007.

[33] A. Silberschatz, H. F. Korth and S. Sudarshan, "Data models," ACM Computing Surveys (CSUR), vol. 28, no. 1, pp. 105-108, 1996.

[34] A. Harth, J. Umbrich, A. Hogan and S. Decker, "Yars2: A federated repository for querying graph structured data from the net," Springer, 2007, pp. 211--224.

[35] Apache, "Apache Jena Elephas," Apache, [Online]. Available: https://jena.apache.org/documentation/hadoop/. [Accessed 10 Feb 2016].

[36] O. Erling and that i. Mikhailov, "Towards web scale RDF," Proc. SSWS, 2008.

[37] S. Fortunato, "Community detection in graphs," Physics Reports, vol. 486, no. 3, pp. 75-174, 2010.

[38] Goldberg, D. Edward et al., Genetic algorithms in search, optimization and machine learning, vol. 412, Addison-wesley Reading Menlo Park, 1989.

[39] M. Tasgin, A. Herdagdelen and H. Bingol, "Community detection in complex networks using genetic algorithms," arXiv preprint arXiv: 0711.0491, 2007.

[40] C. Pizzuti, GA-Net: A genetic algorithm for community detection in social networks, Springer, 2008, pp. 1081-1090.