ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

Managing Big Data Using a Data-Aware HDFS and Evolutionary Grouping System

Dr.P.Baskaran,

Department of Computer Applications, Voorhees College, Vellore, Tamilnadu

Abstract— The increased use of cyber-enabled systems and Internet-of-Things (IoT) led to a massive amount of data with different structures. Most big data solutions are built on top of the Hadoop eco-system or use its distributed file system (HDFS). However, studies have shown inefficiency in such systems when dealing with today's data. Some research overcame these problems for specific types of graph data, but today's data are more than one type of data. Such efficiency issues lead to largescale problems, including larger space required in data centers, and waste in resources (like power consumption), that in turn lead to environmental problems (such as more carbon emission) [1], as per scholars. We propose a data-aware module for the Hadoop eco-system. We also propose a distributed encoding technique for Genetic Algorithms. Our framework allows Hadoop to manage the distribution of data and its placement based on cluster analysis of the data itself. We are able to handle a broad range of data types as well as optimize query time and resource usage. We performed our experiments on multiple datasets generated via LUBM.

Index Terms— Clustering methods, Distributed Computing, Information Management, Optimization, Scalability

1 INTRODUCTION

Building a science out of data faces many challenges. One major problem is that today's data is big, dynamic, and heterogeneous, collected from multiple sources and frequently has no standard structure. The majority of modern data analytics, management tools and services are designed to use Hadoop Distributed File System (HDFS) as a data warehouse; sometimes these analytic tools use services provided by the Hadoop ecosystem for processing. From a price/performance standpoint, Hadoop stands well. We transformed the data and stored it in graph-based scalable stores to give it a sense of

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

structure and to be able to stream changes, constructing vertex-to-vertex triples for data points, then adding cluster affiliation data to these triples to form quadruples as described in the architecture section. These techniques allowed us to (1) collect data from multiple sources and convert them into quads with a sense of structure for different data, (2) stream changes dynamically and push to the graph database, and (3) prepare the data for application to a new version of Hajeer et al. [12]. A novel encoding of chromosomes was used to handle the modern data clustering problem along with novel crossover, mutation and evaluation techniques to deliver the needs of the new distributed encoding technique. Later, we distributed the sub-graphs over HDFS based on the cluster affiliations to produce optimized data to query and process.

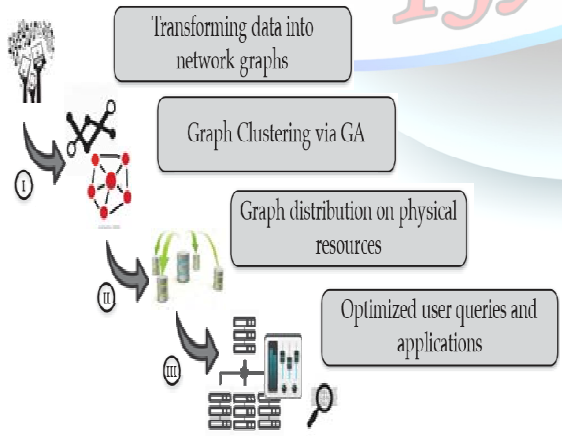

Fig. 1. Computational steps of the proposed framework.

Fig. 1 illustrates the contribution and modules on the proposed framework as follows:

(1) after collecting the data or gathering old datasets, this module converts the data into the desired network graphs;

(2) finding patterns in the graphs, the module distributes the data into the right data blocks;

(3) distributes the blocks into the right machine accordingly; and

(4) an optimized DHFS serves as a data source for services to execute queries and provide a platform to apply graph algorithms efficiently as well as reduce resource usage. To summarize the above, the proposed framework improves the ability of HDFS to handle modern data by building data awareness modules that detect, distribute, and manage data over the scalable file system. Thus, the framework results in optimization and efficient resource usage of the Hadoop eco-system and other tools and services that use HDFS as a distributed storage.Promising solutions in next generation analytics and lambda architecture have been presented in recent studies. Song et al. [13] reviewed the recent research in data types, storage models, analysis methods and application to network Big Data. They also

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

summarized the challenges and development of big data to predict current and future trends.

Service deployments over HDFS

As mentioned before, HDFS serves as a distributed data source for modern big-data solutions, such as Apache Spark [6], Mesos [9], HAMR [8] and hundreds of others. Such solutions have many deployments, mostly over HDFS or over a service that runs on HDFS (see Fig. 3).

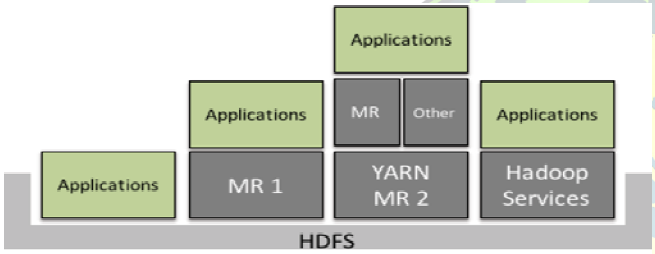

Fig. 2. Services and applications deployments on HDFS.

2. HDFS PERFORMANCE & EFFICIENCY PROBLEM

The utilization of the Hadoop eco-system to process enterprise data and build applications on top of it is dependent upon the enterprise use-cases and the data. Since IT BI teams (business intelligence) in businesses and enterprises configure such systems to meet their goals and roadmaps, they focus on the data and use-cases. Most enterprise data are collected for specific use cases. Later, these data reside on storages waiting for the BI team to make use of them, thus resulting in data collected from multiple sources having multiple structures. As per Huang et al. [1] and Rohloff et al. [5], the implementation of Hadoop and the services that are designed to run on HDFS lack optimization for graphs. Some of the causes for HDFS inefficiency include the following as per [1]: (1) the default hash partitioning provided by Hadoop may lead related data to end up far away physically over the set of computing resources, effectually resulting in a massive amount of data transfer between resources to finish graph operations. Thus, combining related data is a win as per [1]; (2) Hadoop considers the same importance for all data blocks and partitions, so maintaining the locality of inter-cluster neighbors and keeping them physically close-by improves efficiency, and (3) HDFS is not optimized for graph data. Huang et al. [1] showed an efficiency problem with the said technique in Rohloff et al. [5] within a Hadoop-based system. However, the manner in which Huang et al. [1] and Rohloff et al. [5] worked around the problem can be generalized. Since they focus on one particular file type and one simple clustering algorithm, we believe that this technique has some drawbacks when we deal with big and dynamic un/semi/multi-structured data. In a previous study, Hajeer et al. [12], we confirmed such limitations. The study showed how to use genetic algorithms to cluster such data. We used

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

this technique in Hajeer et al. [12], Pizzuti [43] and [42] along with a list of other work, such as [48], [46], [62], and [63], after building a transformation method to convert desired data into graph data. The results of extending the work in [12] were used to generalize Huang et al. [1] and Rohloff et al. [5].In Hadoop, the main idea is to bring the computations to the data; for example, MapReduce, the Map part, can quickly bring the computations to the container that has part of the data as its resource. On the other hand, the reduce phase collects data from the mappers' outputs, and in most cases, these data parts have to travel over the network to the right containers that run some specific reducers. Such mappers' output usually gets collected from multiple mappers to each reducer. Programmers have to control such problems to some extent. That is why even at Cloudera's training for developers they tend to give a guideline for programmers to use mapper side joins rather than reducers. Cloudera's training also encourages developers to use Hive [64] query planner when possible to take care of such joins. Hence, we strongly believe that the Hadoop cluster itself should have a sense of data awareness, where data from mappers' outputs for the same key should be in the same, or at least a nearby, machine as much as possible. Since applications and jobs are different from one usecase to another, it is impossible to cover all cases and future cases. One solution to optimize such jobs is to cluster the data as the proposed framework does. Not all algorithms are covered in such cases, but most graph algorithms specifically rely on the data that are related and connected (e.g. graph traversing).

3. DATA-AWARE HDFS INDICES

3.1 Graph Transformation

We discussed in the introduction the sources and problems of modern data. And we mentioned that data could come from different sources LMN where *S*N is source *Z* and *Z*. These various sources generate or contain data with different structures, sometimes for the same entities but different data and different structures. LM N where is the structure of data coming from source *Z* and contained to the infinite superset *DS* (Structure) ⊂ L∞ Data with the structure ∈*D* were transformed into a graph *G* (V, E) as an undirected graph and with the number of vertices |*V*|=*m* and the number of edges |*E*|=*n*. This transformation is further explained in the Overall Architecture section.

3.2 Graph Clustering

We referred to a graph of vertices V and edges E as *G (V, E)*, as a directed graph. Also, the number of vertices |*V*|=*m*, number of edges |*E*|=*n* and the clustering LM N as a partition of *V* as

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

disjoint sets. We call *C* a clustering of *G* containing *J* clusters. The number of clusters *j* has a minimum of *j=1* when *C* contains only one subset L, and a maximum of *j=m* when every cluster contains only one vertex. We identify the cluster *C*j as a subgraph of *G*. The graph QRLMMNN where MNLOOPLO P Then MNL∪ $^{?}_{730}$%M$Nis the set of intra-cluster edges and *E/E(C)* is the set of inter-cluster edges. The number of intra-cluster edges denoted by *m(C)*and *-(C)* is the number of inter-cluster edges. In our clustering algorithm, we used modularity as a fitness measure in Hajeer et al. [12]. Modularity *Q* is then defined as the fraction of edges that fall within group 1 or 2, minus the expected number of edges within groups 1 and 2 for a random graph with the same node degree distribution as the given network.Hence, the actual number of edges between *v* and *w* minus expected number of edges between them is MN. Modularity can be expressed in Equation

3.3 Graph Distribution and Assumptions

As described previously, three major challenges faced HDFS optimization; two of them were about how Hadoop hashes and distributes the data. Our assumption and experiments showed that (1) storing intra-cluster data together on the same machine and (2) storing close inter-clusters data on close-by machines were a huge step toward optimizing HDFS.

4. OVERALL ARCHITECTURE

Our clustering framework (DEGA-Gen) is a part of the proposed data-awareness module running on top of the distributed data storage as shown in Fig. 4. The framework interacts with HDFS and its available services to provide updated clusters as data flows in HDFS. Our goal is to achieve optimization by placing related data together and reducing overhead on data movement between hosts. Data transfer mostly happens in aggregation processes or joins.

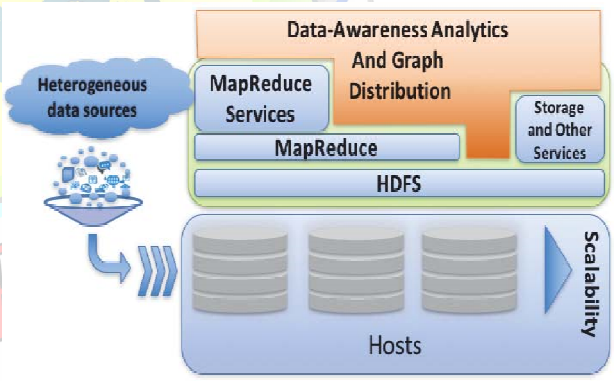

Fig. 3. Data awareness module and Distributed evolutionary clustering algorithm as part of Hadoop.

4.1 Building Distributed RDF Graphs

The first step performed by the data-aware HDFS framework is converting datasets into a distributed RDF graph. This process was done using the open sources Apache Jena and Apache Jena Elephas. The proposed dataaware HDFS

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

turns the datasets into quadruples rather than triples; reasons for this process are explained in the Genetic-Based Clustering. Unlike the widespread use of quadruple representation, we used the extra field in the quads to represent cluster affiliation of the triple rather than the graph membership of the quad; we called the field (Chromosome ID). This process leverages the usage of quad stores to enhance the sub-process (encoding and representation for distributed clustering).

Not all types of data can be transformed into graphs. However, Big Data is about the use-case and the data in some cases. This transformation is direct (SHARD), and relational database records (for example) can be represented as an RDF graph where each attribute of each record is a relation between two nodes, one node represents the value of the key field and the other node is the value of the attribute field. In other cases, (like text data), relations between data points can be defined based on the use case. Take natural language processing, for example, one can define relations of (word comes before) and (word comes after) for each word to build a prediction model.

4.2 Genetic-Based Clustering

4.2.1 Encoding & Representation

We used the encoding from [65] to overcome the big encoding issues found in previous studies and listed in [65]. Such encoding derives from the definition of clusters. However, even with such encoding in [65], solutions can still have a very long representation as the data scales up. Eventually, the GA client will run out of memory handling solutions itself as the data scales up. Another technique we used to reduce the overhead of manipulating solutions was to store it as extra information along with graph triples on HDFS, by converting data points from <Node><Predicate><Node> triples, as in Rohloff et al. [5], into <Chromosome_part><Node><predicate><Node> quadruples. We referred to <chromosome_part> as a list of solution_IDs that this particular node belongs to in the population. This encoding leads to a population of a fixed size list of Integers on the GA client side called solution_IDs. This technique allows the client to scale the clustering GA on larger size datasets that the HDFS can hold.

The idea was to treat solutions as data and to inherit all scalability properties that apply to the graph. Thus, the population of a size X on the client side has a constant size(X) regardless of the data size. We referred to this novel technique as Distributed chromosomes, and as a concept, it is about the distribution of genes from the solutions along with the data. Fig. 5 explains how the graph

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

data were stored in RDF format and how we performed the integration of solution encoding on RDF data. We used Apache Jena and Jena Alephas and modified these open sources to match our needs. Convert_to_quads_Chromo class was developed to convert RDF graph Triples to Quads as in Fig. 5. This class contained Mapper, reducer, combiner, and appropriate writables as well as input and output classes formatted to deal with RDF data. It takes each triple from each block of data and converts it into a quad with a random gene (part of solutions) that it belongs to then stores it back into HDFS.

4.2.2 Objective Functions

In our clustering algorithm, we maximize modularity as an objective in Hajeer et al. [65]. As per [57], Modularity Q is defined then as the fraction of edges that fall within group 1 or 2, minus the expected number of edges within groups 1 and 2 for a random graph with the same node degree distribution as the given network.Hence, the actual number of edges between *v* and *w* minus expected number of edges between them is *Avw*-(*kvkw*)/2*m*. Please refer to Equations 1-5 in the Data-aware HDFS Indices section. Note that modularity maximization is not the only objective. Another objective is to minimize the solution length. Considering intra-cluster edges as inter-cluster edges results in some longer solutions with no difference in modularity. Hence, those solutions need to be given a smaller fitness but not totally ignored (a combination with other solutions may lead to a better clustering). Since our evaluation on the datasets used considers predicates and relations to work both ways (an undirected graph), we used modularity in Equation (3). For use cases where the defined relationships result in a directed graph, there is an extended modularity that was proposed for a directed graph that can be utilized. On the other hand, the clustering purpose is to find a better placement for graph data. Hence, considering directed graphs as undirected graphs while clustering does not necessarily force the user to have the same assumption while querying the data.

4.2.3 Details Steps of Genetic Clustering Population Initialization

Population initialization is the process of creating a collection of diverse solutions. As described in the Encoding & Representation sub-section, we transform the triples in RDF data into quads, adding the ability to hold a gene (part of the solution) for each vertex, where this gene is a list of random solution IDs to which a particular vertex belongs. For example, if a Quadruple *D* has *S1* and *S5* as genes, then that is translated as solutions *S1* and *S5;* both consider the edge *D* as

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

an inter-cluster edge connecting two separate clusters. In ∀*TiT* there is a set of solutions ∪*STi*"(, where *T* is ∪ of triples *t* that constructs the graph *G,* and *S* is the set of solutions *s* in the population. It is critical to keep in mind that the maximum size of such a list is the integer size of the population. The initialization process for populations is shown in Fig. 5. The GA client only holds a two-dimensional array of solution-IDs and Modularity Fitness (Integers and floats), allowing the client to start the selection process and initiate the distributed GA operators. Working on a fixed, small size two-dimensional array, where the real genes are stored in the data blocks in a distributed manner taking advantage of HDFS, proved to provide more scalability.

5. Solutions Evaluation

The evaluation was done using the objective functions described in the objectives section. Each solution is evaluated by computing modularity on the analogous graph, a graph where edges in the solution are marked as inter-cluster edges. We identified the clusters by removing the marked edges and considering the disconnected graph components as communities. Then, we computed the modularity considering the marked edges again as intercluster edges. The process of computing the modularity on a large graph is both resource and time consuming, so we decided to improve it using distributed tasks to be run on the quadruple store created with extra data for solutions. Using HDFS and distributing the dataset over multiple machines, we were able to batch process each set of solutions (generation) at once.

Given a Population_ID list S that contain ID's of solutions to be evaluated

---

```
Map(Key index, Value Quad):
  ForEach solutionID in S:
    If Quad.GetGenes in solution:  Quad.marked = True  Emit (solutionID , Quad)
    Else:
      Quad.marked = False
```

Fig. 4. Distributed evaluation of solutions (map task).

After the client side of the algorithm injects current population solutions data into the quadruples stored in HDFS, it sends the list of solution IDs (list of integers) to be evaluated. Fig. 6 illustrates the evaluation Map tasks. The map function is called for each Quad in the graph chunk that represents part of the graph. Jena Elephas is used with modified input and output

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

class to use Chromosome quads rather than default graph quads. Each container on the HDFS cluster performs a map operation on the graph chunks it has been assigned. After mapping all the chunks into pairs of <Keys, Values> representing solution IDs and Quads that are part of the corresponding solution, the shuffling task takes place. All values for the same key are grouped together as <Key, List of Values> that represent each solution and the list of marked and unmarked Quads (Graphs where inter-cluster edges are marked). The final stage consists of the reduce tasks that are described

Given a solution S and a set of Quads marked based on S, as mappers' outputs and reducers' input for a graph G with N Quads

Reduce *(Solution S, EdgesQuads [E1,E2,E3,….EN]):*
ForEach *Quad E* ***in*** *QuadssList:*
If*E.marked =* ***True****:*
MarkedQuads.append(E)
Else*:*
UnMarkedQuads.append(E)
Endfor
Communities = ***FindComponents****(MarkedQuads, UnMarked-Quads)*
Modularity = 0

ForEach *Community C* ***in*** *Communities:*
M46@@<A5;=<B!1249DCC<A5;=<BN
DegreeFraction
=_____
M1!8N
Modularity += (C. InnerEdges /N)-(DegreeFraction)^2 ***Endfor***
Emit*(S, Modularity)*

Fig. 5. Distributed evaluation of solutions (reduce task).

The FindComponents function was implemented using a modified linear finding component algorithm to store also the number of intra-cluster edges and the number of inter-cluster edges for each community. When reduce tasks finish, the results of reducers are written to HDFS, and the results contain each solution with its modularity. The results consist of a fixed size two-dimensional array of integer solution IDs and a fitness for each solution. The evolutionary algorithm reads this file and continues working on an evaluated generation ready for selection, crossover and mutation processes. In the last generation, an extra piece of information controlled by a boolean configuration variable is written to HDFS as well; this piece contains the clustering affiliation for each node. The reason they are only written in the last generation is to

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

lower the write overhead on HDFS while affiliations are not needed any time before it.

6. Crossover, Mutation and Selection

Since we stored the chromosomes in a distributed manner, we needed to modify the GA operators used in Jmetal open source to be able to run them on the corresponding quadruples that represent the graph. This procedure was done by developing distributed crossover and distributed mutation modules, which in return created jobs of crossover and mutations to be performed on the corresponding population. After evaluating the population, the selection process starts based on each solution ID and its fitness. Tournament selection is the selection used, and the reason is to avoid converging to locally optimal solutions, which are a lot based on our encoding technique. By ranking the population and choosing solutions from each class, a set of parents along with the new offspring IDs were constructed. Fig. 8 is a high-level diagram showing the steps in which the algorithm creates GA operator's tasks.

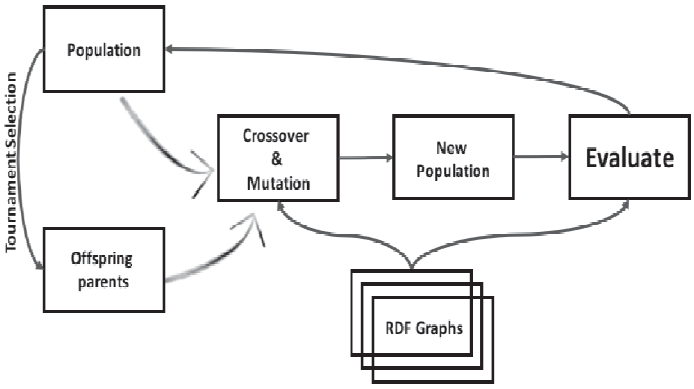

Fig. 6. Distributed genetic algorithm for RDF clustering.

Encoding and storing solutions in a distributed manner delivered the advantage of small and fixed size populations on a client side. However, GA operators in the open source Jmetal needed to be modified as well as NSGAII, which was used in our case. The original NSGAII creates a population's and offspring's solutions and evaluates it one solution at a time; such a situation creates an overhead of tasks on HDFS. Rather, we modified NSGAII to DNSGAII (Distributed NSGAII) by creating a set of solutions then performing evaluation and GA operators at once in one MR2 task. Fig. 9 illustrates the task of distributed crossover and distributed mutation. The distributed crossover and mutation task takes the population as inputs along with the selection results, then for each quad in the data changes the partial chromosomes accordingly. The task removes any solution ID (gene) that does not belong to the current population to save space and

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

computations. Then, as shown in Fig. 8, the new offspring population is sent to evaluation. Here we have to note that solutions that belong to a previous generation will not be evaluated since they already have fitnesses. This copy technique of fitnesses saved an enormous amount of computations when we dealt with big data for a long series of generations. The processes of representation, population initialization, evaluation, selection and offspring evaluation to population are illustrated in Fig. 10. The numbers represent the processes and tasks order. Since we were dealing with dynamic data as one of the Big-Data five V limitations (Velocity, Variety, Veracity, Value and Volume), the algorithm gets suspended when it converges to the same solution for a sequence of generations then continues working as new data arrives to start from the last generation reached.

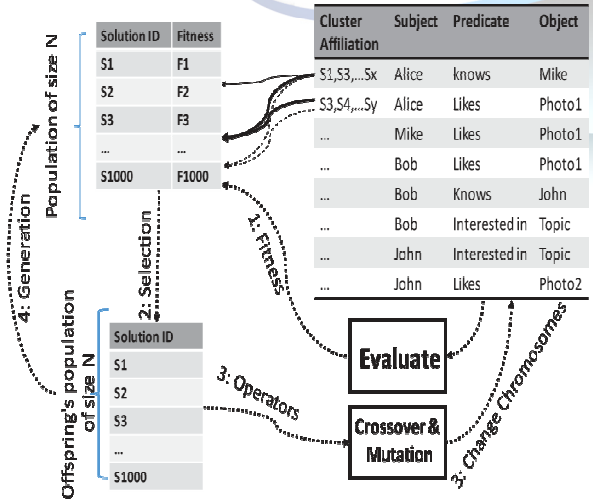

Fig. 7. Distributed genetic algorithm clustering process flow.

Suspension of the algorithm ensures that clustering will apply to the new data while old data have the best affiliation found, so there is no need for the clustering process to start from the beginning.

7. Partitioning and Placement

After clustering the RDF data, the last step was to repartition the data and place graph quadruples accordingly. The goal in this step was to place quadruples that belong to the same cluster and have a high degree of connectivity into the same partition to ensure locality of intra-cluster quadruples. Another goal was to place highly connected inter-clusters into a close partition physically, to map the inter-cluster distance onto the physical distance of partitions. Fig. 11 illustrates the desired allocation of quadruples, assuming that the horizontal distance in the figure represents the physical distance between the computing nodes (the distance of network routing). We account for the distance of HDFS nodes by how many routing hops between them (networks, routers, switches…). We set up HDFS over machines connected using multiple networks to create a distance in routing. The placement script placed quadruples. [7] discussed about a Secure system to Anonymous Blacklisting. The

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

secure system adds a layer of accountability to any publicly known anonymizing network is proposed. Servers can blacklist misbehaving users while maintaining their privacy and this system shows that how these properties can be attained in a way that is practical, efficient, and sensitive to the needs of both users and services. This work will increase the mainstream acceptance of anonymizing networks such as Tor, which has, thus far, been completely blocked by several services because of users who abuse their anonymity. In future the Nymble system can be extended to support Subnet-based blocking. If a user can obtain multiple addresses, then nymble-based and regular IP-address blocking not supported. In such a situation subnet-based blocking is used. Other resources include email addresses, client puzzles and e-cash, can be used, which could provide more privacy. The system can also enhanced by supporting for varying time periods.

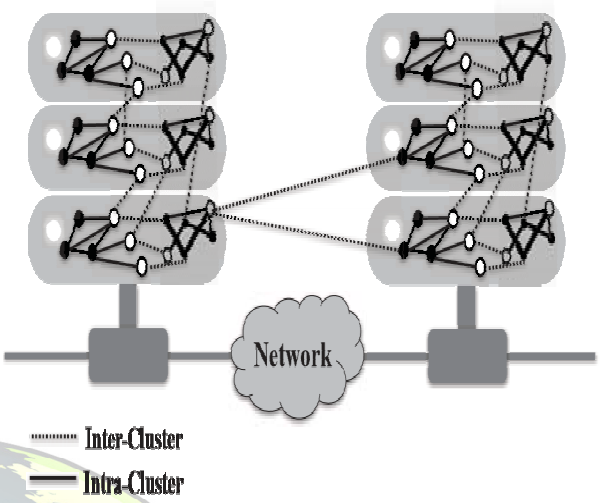

Fig. 8. Quadruples placement in different HDFS nodes.

Partitions are created based on the number of machines; each machine has its own partition. A MapReduce job scans the quadruples and places quadruples related to one random cluster in one partition then emits the placed quadruples, leading to where all connected clusters' IDs are stored for the next scan. The second scan places the quadruples for the closest inter-clusters in the same partition and emits from the original dataset. Further, the second closest inter-clusters are placed into the next closest partition. When there are no more interconnected clusters left, another random cluster is chosen from the dataset until no more data is available. Fig. 11, for example, illustrates how clusters with three inter-cluster connectivity are placed in the same HDFS node, and with two inter-cluster connectivity are placed

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

in the next HDFS node, further clusters are placed in further nodes.

8. EXPERIMENTS AND RESULTS

We divided the experiment into two sections: the first section is the design and the testing of the clustering algorithm on the graph store to test the clustering results, and the second section is about the tests and comparisons of the effect of optimization framework on HDFS. All graphs and trend models are processed using Tableau [66].

9. System Performance Experiments

In this section, we measure the performance of our system against multiple RDF stores including SHARD after clustering and placement. We used Cloudera Impala to create a table on the data. We focused mainly on time and resource usage.

10. Clustering and Load Time

The results of loading 270 million RDF triples into a twenty-machine HDFS cluster as per Huang et al. [1]. For a ten-machine cluster, as per Alexzander et al. [24], the loading time was 40 minutes. Because of the differences in resources, for benchmarking queries in the Query Performance Comparison section, we normalize results to be able to compare query response time. There is a noticeable workload in terms of time to prepare the Cluster-based partitioned RDF against hash partitioning the data. However, the effect on optimization during query time and storage is a trade off, as we describe in the Query Performance Comparison section. It is critical to point out that the number of triples has a larger effect than the storage size in GB; since different compression can solve the storage size issue but not the amount of information needed to be processed. It is also very important to note that clustering results are stored by our proposed architecture, so when new data and changes become available, the algorithm does not need to start over. In other words, adding changes to data, to some extent, happens in close to real time by adding the new triples to the physical server that has the data-cluster it connects to.

11. CONCLUSION

In this article, we presented a data-aware HDFS and the services running on top of HDFS that optimize state-ofthe-art RDF stores. We proposed a cluster-based data partitioning to manipulate the physical locality of the data to match the graph locality as well as the causality in HDFS processes. This allowed parallel processing of queries for data on HDFS that required less resource usage. Our framework was able to perform faster than some attempts and slightyly slower than other attempts for scalable RDF data stores. However, with less resource usage. Studies in next-generation analytics and lambda

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

architecture [15], [16], [17] and [18], along with Apache Kudu [20] and a set of studies in [21] proved to be fast and more efficient in processing of OLAP workloads and showed a strong performance in running time-critical workloads. It is worth the effort, however, to study the impact of intelligent data placement on such methods. For future work, we plan to further improve the distributed encoding and the genetic operators to reduce computation overhead. We also plan to experiment with dynamic updates for a larger velocity of data flow and to utilize tools and frameworks of the lambda architecture and next-generation analytics presented in the recent studies.

12. REFERENCES

[1] J. Huang, D. J. Abadi and K. Ren, "Scalable SPARQL querying of large 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 data warehousing queries in a split execution environment," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011.

[3] M. Walker, "Data Science Central," 19 Dec 2012. [Online]. Available: http://www.datasciencecentral.com/profiles/blogs/structure d-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 biggest growth in the far east," *IDC iView: IDC Analyze the future,* vol. 2007, no. 2012, pp. 1--16.

[5] K. Rohloff and R. E. Schantz, "Clause-iteration with MapReduce to scalably query datagraphs in the SHARD graphstore," in *Proceedings of the fourth international workshop on Dataintensive distributed computing*, 2011.

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

[7] Christo Ananth, A.Regina Mary, V.Poornima, M.Mariammal, N.Persis Saro Bell, "Secure system to Anonymous Blacklisting", International Journal of Advanced Research in Biology, Ecology, Science and Technology (IJARBEST), Volume 1,Issue 4,July 2015,pp:6-9

[8] .E. I. Inc., "HAMR - Faster than the speed of data," ET

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

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.

[11] 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.

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

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] M. Congosto, P. Basanta-Val and L. Sanchez-Fernandez, "THoarder: A framework to process Twitter data streams," *Journal of*

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 4, February 2018

Network and Computer Applications, vol. 83, pp. 28--39, 2017.

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

[21] N. Marz and J. Warren, Big Data: Principles and best practices of scalable realtime data systems, Manning Publications Co., 2015.

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