

MASSIVE DATA PROCESSING USING MAPREDUCE AGGREGATION TO MAKE DIGITIZED INDIA

¹S.Thilagavathi, ²S.Vimala, ³K.Valarmathi, ⁴R.Priya, ⁵S.Sathya

¹PG Scholar, Computer Science and Engineering, Panimalar Engineering College, Chennai, India

²Associate Professor, Department of Computer Science and Engineering, Panimalar Engineering College, Chennai, India

³Professor, Department of Computer Science and Engineering, Panimalar Engineering College, Chennai, India

^{4,5}Assistant Professor, Department of Computer Science and Engineering, Panimalar Engineering College, Chennai, India

thilagavathisivaraj@gmail.com, vimalakumaran@gmail.com, valaryogi1970@gmail.com, priyarsp@gmail.com, sathya27489@gmail.com

Abstract - Digitized India is used to connect rural areas with high speed Internet. As a result, it is used to reduce crime, manual power, documentation and also increases the job opportunities. Nowadays people are facing many problems when they forget to carry the driving license and also to reduce the corruption, the proposed system combines the driving license with Aadhar card. The details of driving license and Aadhar card data can be combined using the MapReduce Counters. It automatically aggregated over Map and Reduce phases. It is used to create a tool that manages the handling of license using unique identification associated with each individual. It helps the user to travel various places without having the license. So the proposed system will make the digitization of data on a large scale for easy and quick access throughout the India. Sqoop is a tool intended to exchange information amongst Hadoop and social databases. Sqoop utilizes MapReduce to import and export the information, which gives parallel operation and in addition adaptation to non-critical failure. As the result of parallel operations time utilization for transferring the data get decreased radically.

Index Terms - Digitized India, data skew, MapReduce, Sqoop.

I. INTRODUCTION

Huge Internet organizations routinely create many tera-bytes of logs and operation records. MapReduce is a programming model for processing large data set in distributed and parallel processes stored inside the Hadoop distributed file system [13]. Map Reduce has ended up being a capable gadget to process such broad educational indexes. Map Reduce has been generally utilized as a part of different applications, including web ordering, log examination, information mining, logical reproductions, machine interpretation, and so on [7]. There are a few parallel processing systems that help Map Reduce, for example, Apache Hadoop, Google Map Reduce, and Microsoft Dryad, of which Hadoop is open-source and generally utilized [7].

Hadoop is an open source framework for processing and analysing of big data with the help of HDFS and MapReduce. The traditional database is stored in an RDBMS like Oracle, MS SQL Server or DB2 and a enhanced and sophisticated software will be written to interact with the database, process the desired data and present it to the users for the purpose of analysis[8].

Apache Hadoop is developed for not only structured datasets but it can also process unstructured datasets. NoSQL database has turned into a popular distributed database framework that pulled in numerous considerations among endeavours and scientists. Database engineers in many organizations consider about the movement of relational

databases to NoSQL databases for the effectiveness of taking care of enormous information.

NoSQL databases have emerged as a solution to the aforementioned drawbacks and have become the preferred storage option for big data applications. Currently, there are more than 225 recognized NoSQL databases [18]. The basic operations in a database can be formulated from one or more of the following: Create, Read, Update and Delete (commonly referred as CRUD). Data stores can be tailored to handle varied workloads of CRUD operations to satisfy the requirements of specific applications. Therefore, it is necessary to identify, among the available databases, the optimal NoSQL database for a given application workload.

Apart from the Hadoop services, the Hadoop Ecosystem also includes various other tools as per the particular requirements. The other tools which are part of the ecosystem are namely Hive, Pig, Flume, Zookeeper, HBase etc [17]. Hive is a data distribution centre programming venture based over Apache Hadoop for giving data rundown, request, and investigation. Hive gives a sql like interface to address informational collection away in various databases and record structures that wire with hadoop. Traditional SQL request must be completed in the MapReduce Java API to execute SQL applications and inquiries over scattered data. Hive gives the vital SQL reflection to arrange SQL-like inquiries (HiveQL) into the essential Java without the need to implement queries in the low-level Java API.

Here, Sqoop underpins incremental heaps of a table or SQL queries and additionally spared occupations which can be run various circumstances to import refreshes made to a database since the last import. Imports can likewise be utilized to populate tables in Hive or HBase. Sqoop got the name from sql+hadoop. Sqoop import and export tools are used to import and export the data.

In this paper we address the issue of effectively handling MapReduce occupations with complex reducer undertakings over skewed information. The information skew issue in MapReduce has been contemplated. When MapReduce keeps running in a virtualized cloud registering environment, for example, Amazon EC2, the registering and capacity assets of the hidden virtual machines (VMs) can be differing for an assortment of reasons.

II. RELATED WORKS

To handle efficient voluminous data the on-process aggregation [5] of Map Reduce was used to amend the performance of Hadoop. It acts as an interface between the

name node and data node to reduce the burden of name node and to improve the performance of the data being processed. Compute-aggregate tasks [6] allows to identify unify design principals of "ideal" aggregation trees. This approach has the preferred standpoint of simple recovery at the time of disappointments; in any case, up to all mappers have finished reducers can't begin executing their assignments. It increases the utilization and reduces the response time. As a result it increases the performance of the system.

In [2], [3] distributed algorithm is design to improve the performance of network traffic. The performance of the MapReduce is improved by increasing the performance of the network traffic in shuffle phase. To decrease network traffic inside a MapReduce job, we need to consider total information with comparable keys some time recently sending them to remote reduce tasks. Despite the fact that we have a comparative capacity, called combiner, which has been as of now received by Hadoop, it works quickly after a map task exclusively for its produced, neglecting to neglect the information aggregation opportunities among various tasks on various machines.

Block chain query processing algorithm [8] was used to overcome the skew and to improve the overall execution time of the assigned job in the system. In order to overcome skewed distribution, one of the evolutionary models named spark [9] were used to balance the dataset by selecting the most relevant data. Data Skew is one of the most significant issues in MapReduce applications and hence LIBRA [10] was used.

The imbalance in the amount of data assigned to each task causes some tasks to take much longer to finish than others and can significantly impact performance. In Traditional Hadoop the information is separated into partitions on a static way that each slave node will be apportioned with approach size of information which may prompt the Data Skew problem.

III. EXISTING SYSTEM

A. Network Traffic

The programming model called MapReduce rearranges vast scale information handling on item group by running map task and reduce task simultaneously. Most existing work dedicated on MapReduce performance improvement by optimizing its data transmission and they turn out to be very near each other, which indicates that distributed algorithm can be appropriate in practice [2]. The MapReduce job is separated into different time slots with a length of several minutes or an hour. When compare to existing method in Hadoop, the proposed system reduces the network traffic by placing the aggregation in the shuffle phase [2].

To reduce network traffic within a MapReduce job, we need to consider aggregate data with similar keys before sending them to remote reduce task. Objective is to limit the aggregate system activity by Data segment and accumulation for a MapReduce work. The data partition and aggregation in a dynamic manner is designed by another method called online algorithm. Another additional challenge which emerges in managing the mapreduce work is for big data. To solve the problem on many machines in simultaneous manner we use

distributed algorithm. The main essential thought behind this is to break down the first vast scale issue in to a few dispersed feasible sub problems that are facilitated by a high-level master problem. In [2],[3] distributed algorithm is design to improve the performance of network traffic. Neighbor approach on Hadoop platform was proposed in order to predict traffic flow by using pre processing technique[4]. Today the traffic data has been entered and erupted the time of huge transportation of the data. Hence it is important to predict the traffic information.

B. Data Skew In Mapreduce

. Each task is assigned with data which is in imbalance amount causes some tasks to take longer time where some task finish in shorter time which leads to reduce the performance. LIBRA, a lightweight methodology gives solution for the information skew issue in mapreduce[10]. In Hadoop bunches the information is partitioned into segments on a static way that every individual slave nodes will be dispensed with parallel size of information, which might prompt the Data Skew issue. As all slave nodes are not similarly proficient and furthermore the multifaceted nature of information won't likewise be comparative and a few slave nodes may process the errand allotted for a longer time than the other slave hubs in the group.

Map node will search up for the finishing of all Map Tasks and in this situation will sit tight for the long running errand on the slave node to finish. This time slack should be dispensed so that the master node will increase the overall output. Block chain query processing algorithm [8] was used to overcome the skew and to improve the overall execution time of the assigned job in the system. In order to solve skewed distribution, one of the evolutionary models named spark [9] were used to balance the dataset by selecting the most relevant data.

C. Analysis of NoSQL Databases

NoSQL database are used to process a high volume of different data with a great velocity. But this data does not support transaction. Key-value databases, column stores and graph databases are different forms of inputs to a database. NoSQL database implements the concept of 3Vs such as Volume, Velocity and Variety. Volume represents the large amount of data, Velocity represents the speed of the data being processed and Variety represents the different type of data being processed. In each of the implementations, NoSQL databases (Riak and MongoDB) act as a stable storage for the user's application (i.e. application data is stored persistently in the NoSQL database)[16].

There are different types of databases to process the unstructured data such as HBase, Pig, Hive and Cassandra. HBase database is used to handle large amount of database such as terabytes to petabytes of data. HBase is a column oriented database which stores the data in the form of table. Two or more column combined to form column families. It represents the data in the form of key-value pairs. Online Analytical Processing is done in column-oriented database. Cassandra is a combination of key-value and column-oriented databases. It supports transaction and provides scalability. Pig is a dataflow language which is simple to understand. Pig interpreter automatically convert the pig script to the

MapReduce jobs. So it's not necessary that we should possess much knowledge about it. Hive is used to process the relational database. It stores the relational database in meta store as a metadata.

IV. PROPOSED SYSTEM

A. Sqoop

In order to transfer the data between Hadoop and relational database server a tool called Sqoop is used. Sqoop is used to import the data from the relational database to the HDFS in Hadoop. At the time of importing data to HDFS, it sends the request to relational database to obtain the information of the metadata. Thus the relational database sends the response for it and based on metadata information Sqoop generates java classes. It also exports the data from Hadoop HDFS to relational databases. Thus it automatically imports and exports the data.

Sqoop is a command-line interface application. Sqoop Connectors are component which is used to overcome the challenges of data transfers across the system. There is a JDBC connector to connect the different database and it is supported by JDBC protocol. Sqoop also contains third party connector to connect to the databases. Sqoop have an advantage of fault tolerance and compress the data by using deflate algorithm. It also has the capacity of incremental load which is used to update the table. Sqoop imports data in parallel from most databases.

B. Comparison of Single Mapper and Multiple Mapper

At the time of importing it uses one mapper to import the data. The way HDFS has been set up, it separates expansive documents into huge blocks (for instance, estimating 128MB), and stores three duplicates of these blocks on various nodes in the group. The Application Master send demand to the NameNode, where the reproductions of the required information blocks are stored. In file blocks by utilizing the area information, the Application Master send solicitations to the Resource Manager to process particular blocks on the slave node where they're stored.

Hadoop utilizes a logical representation of the information stored in file blocks, known as input splits[14]. At the point when a MapReduce work customer computes the info parts, it makes sense of where the main entire record in a block starts and where the last record in the blocks ends. Thus multiple mapper does the work but as it uses many map simultaneously it reduces the time consumption.

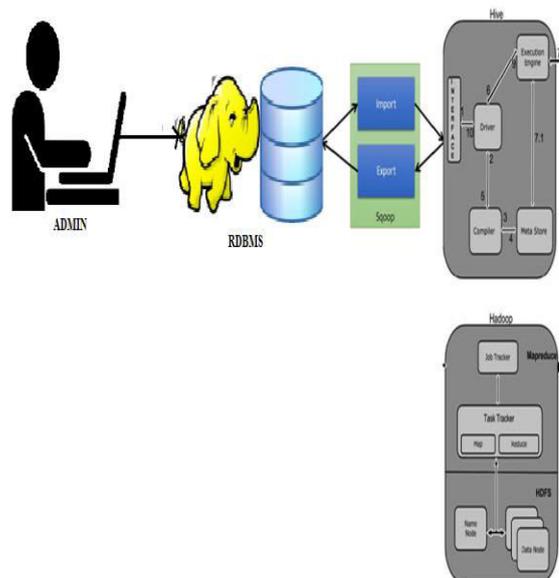


Fig. 1 Architecture diagram

B. Aggregation of Data

To handle efficient voluminous data the on-process Aggregation [5] of Map Reduce was used to amend the performance of Hadoop. It also increases the response time and decrease the response time. Universal characteristics of compute-aggregate tasks [6] allows to identify unify design principals of "ideal" aggregation trees. In order to achieve the aggregation combiner is used which preserves the fault tolerance. Combiner also works as a local reducer which gives the result as soon as possible. Combiner run after the map function and before the reduce function. The yield of such a intermediate outcome diminish activity is called as a snapshot. On-process Aggregation is predicated on usage of on-process Aggregation of Map Reduce in Hadoop for proficient sizably voluminous information handling [5].

In the basic architecture of HADOOP MapReduce framework name node plays an important role during MapReduce programming, the basic functions of name node are to collect the request from the client assign the datasets among the data nodes, update the metadata table records, always keep track of the data nodes, and dispatch the processed output to the client and so it is consider as a master node for the HADOOP MAPREDUCE framework[7]. Aggregator is used to reduce the burden of the name node which acts as the intermediate between the name node and the data node.

One of the major advantages of introduction of an aggregator node in the HADOOP framework would be reduction of major responsibility as well as load of the name node for distribution of task among the data nodes and tracking of task completion through task tracker and the idea is to make the architecture free of bottleneck generated by the single name node as a single point of failure for the whole architecture [7]. Thus the proposed system aggregates the data of two or more Government services such as Aadhar card and driving license.

V. RESULT AND ANALYSIS

Big data is mainly used for processing a large amount of data. Thus time taken for one mapper and multiple mapper is shown below.

```

cloudera@localhost:~
File Edit View Search Terminal Help
17/10/15 22:51:24 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=69235
17/10/15 22:51:24 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=0
17/10/15 22:51:24 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
17/10/15 22:51:24 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
17/10/15 22:51:24 INFO mapred.JobClient: Map-Reduce Framework
17/10/15 22:51:24 INFO mapred.JobClient: Map input records=41
17/10/15 22:51:24 INFO mapred.JobClient: Map output records=41
17/10/15 22:51:24 INFO mapred.JobClient: Input split bytes=869
17/10/15 22:51:24 INFO mapred.JobClient: Spilled Records=0
17/10/15 22:51:24 INFO mapred.JobClient: CPU time spent (ms)=2610
17/10/15 22:51:24 INFO mapred.JobClient: Physical memory (bytes) snapshot=601141248
17/10/15 22:51:24 INFO mapred.JobClient: Virtual memory (bytes) snapshot=4159733760
17/10/15 22:51:24 INFO mapred.JobClient: Total committed heap usage (bytes)=298844160
17/10/15 22:51:24 INFO mapreduce.ImportJobBase: Transferred 5.958 KB in 29.0469 seconds (210.0395 bytes/sec)
17/10/15 22:51:24 INFO mapreduce.ImportJobBase: Retrieved 41 records.

```

Fig. 2 Time Taken for One Mapper

Figure 1 represents the aggregation of Aadhar details and driving license details by eliminating the data skew problem. In this approach it takes a data for processing the query which takes a time of 69235ms for processing large amount of data.

```

cloudera@localhost:~
File Edit View Search Terminal Help
17/10/15 22:06:43 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=8490
17/10/15 22:06:43 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=0
17/10/15 22:06:43 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
17/10/15 22:06:43 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
17/10/15 22:06:43 INFO mapred.JobClient: Map-Reduce Framework
17/10/15 22:06:43 INFO mapred.JobClient: Map input records=41
17/10/15 22:06:43 INFO mapred.JobClient: Map output records=41
17/10/15 22:06:43 INFO mapred.JobClient: Input split bytes=87
17/10/15 22:06:43 INFO mapred.JobClient: Spilled Records=0
17/10/15 22:06:43 INFO mapred.JobClient: CPU time spent (ms)=430
17/10/15 22:06:43 INFO mapred.JobClient: Physical memory (bytes) snapshot=100270080
17/10/15 22:06:43 INFO mapred.JobClient: Virtual memory (bytes) snapshot=693288960
17/10/15 22:06:43 INFO mapred.JobClient: Total committed heap usage (bytes)=49807360
17/10/15 22:06:43 INFO mapreduce.ImportJobBase: Transferred 5.958 KB in 14.9204 seconds (408.9036 bytes/sec)
17/10/15 22:06:43 INFO mapreduce.ImportJobBase: Retrieved 41 records.
[cloudera@localhost ~]$

```

Fig. 3 Time Taken for Multiple Mapper

Figure 2 represents the query processing for the same amount of data by using multiple mapper. In this approach it effectively reduces the time consumption of processing the query by using multiple mapper. As a result it takes only 8490ms.

It clearly shows the time difference between two different approaches for processing the same query. It shows that the proposed system has greater time efficiency. It also stores the map task output locally on the slave machine.

VI. CONCLUSIONS

The aggregation of Aadhar card and driving license detail has been done. Hadoop cluster has been designed and imported using sqoop tool and its effectiveness is compared using multiple mapper algorithm and single mapper Algorithm. Counters are very useful especially when evaluating some MapReduce programs. MapReduce Counters are consequently accumulated over Map and Reduce stages, it is one of the least

demanding approach to examine inward practices of MapReduce programs. This paper also implements the MapReduce Algorithm which stores the Map task yield locally on the slave machine and uses reserve based rendering of results.

The application incorporates different Government offices details into a solitary purpose of Contact. The application can be stretched out to all the Government office details with alteration. New modules can be included without influencing the current modules.

REFERENCES

- [1] Q. Chen, J. Yao, and Z. Xiao, "LIBRA: Lightweight Data Skew Mitigation in MapReduce," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 26, September 2015.
- [2] Joan Mneney, Jean-Paul Van Belle, "On Traffic-Aware Partition and Aggregation in MapReduce for Big Data Applications", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 27, no. 3, pp. 818-828, 2016.
- [3] Priya Gawande and Nuzhaft Shaikh, "Improving Network Traffic in MapReduce for Big Data Applications", *Int. Conf. on Electrical, Electronics, and Optimization Techniques*, pp. 2979-2983, 2016, (DOI: 10.1109/ICEEOT.2016.7755246).
- [4] U.Abirami and S.Sridevi, "Traffic Flow Prophecy With Mapreduce Job For Big Data Driven", *Int. Conf. on Advanced Computing*, pp. 13-18, 2016, (DOI:10.1109/ICoAC.2017.7951737).
- [5] U.Abirami and S.Sridevi, "Traffic Flow Prophecy With Mapreduce Job For Big Data Driven", *Int. Conf. on Advanced Computing*, pp. 13-18, 2016, (DOI:10.1109/ICoAC.2017.7951737).
- [6] William Culhane, Kirill Kogan, Chamikara Jayalath and Patrick Eugster, "Optimal Communication Structures for Big Data Aggregation", *IEEE Conf. on Computer Communications*, pp. 1643-1651, 2015, (DOI: 10.1109/INFOCOM.2015.7218544)
- [7] Bihudutta Jena, Mahendra Kumar Gourisaria, Siddharth Swarup Rautaray and Manjusha Pandey, "Improvising Name Node Performance By Aggregator Aided HADOOP Framework", *Int. Conf. on Control, Instrumentation, Communication and Computational Technologies*, pp. 382-388, 2016, (DOI:10.1109/ICCICCT.2016.7987978)
- [8] Sankari Subbiah, Sathya Mala and Senthil Nayagam, "Job Starvation Avoidance with Alleviation of Data Skewness in Big Data Infrastructure", *Int. Conf. on Computing and Communications Technologies*, pp. 1857-1866, 2016, (DOI:10.1109/ICCCT2.2017.7972264).
- [9] Triguero, M. Galar, H. Bustince and F. Herrera, "A First Attempt on Global Evolutionary Undersampling for Imbalanced Big Data", *IEEE Congress on Evolutionary Computation*, pp. 2054-2061, 2017, (DOI:10.1109/CEC.2017.7969553).
- [10] Qi Chen, Jinyu Yao and Zhen Xiao, "LIBRA: Lightweight Data Skew Mitigation in MapReduce", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 26, no. 9, pp. 2520-2533, 2015.
- [11] Hong Zhang, Hai Huang and Liqiang Wang, "MRapid: An Efficient Short Job Optimizer on Hadoop", *IEEE International Parallel and Distributed Processing Symposium*, pp. 459-468, 2017, (DOI: 10.1109/IPDPS.2017.100)
- [12] Preeti Narooka and Sunita Choudhary, "Optimization of the Search Graph Using Hadoop and Linux Operating System", *Int. Conf. on Nascent Technologies in the Engineering Field*, 2017, (DOI: 10.1109/ICNTE.2017.7947886).
- [13] Subhash Chandra and Deepak Motwani, "An Approach to Enhance the Performance of Hadoop MapReduce Framework for Big Data", *Int. Conf. on Micro-Electronics and Telecommunication Engineering*, pp. 178-182, 2016, (DOI: 10.1109/ICMETE.2016.64)
- [14] SandeepKumar Hegde and Srinivasa K.G, "A Novel Pattern Classifier Approach towards the Performance Optimization of Big Data Analysis in Distributed Environment", *IEEE Int. Conf. on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics*, 2017, (DOI:10.1109/AEEICB.2017.7972391)
- [15] Umüt Klzgmder, Silleyman Eken and Ahmet Sayar, "MapReduce Based Scalable Range Query Architecture for Big Spatial Data", *IEEE Int. Conf. on Computer Systems and Applications*, 2016, (DOI: 10.1109/AICCSA.2015.7507268)
- [16] J. María Teresa González - Aparicio, Adewole Ogunyadeka, Muhammad Younas, Javier Tuya and Rubén Casado, "Transaction processing in consistency-aware user's applications deployed on

- NoSQL databases”, Springer, pp. 1-18, 2017, (DOI:10.1186/s13673-017-0088-3).
- [17] Karan Sachdeva, Japtej Singh Lamba, Vishal Sinha, Neetu Singh, "Comparison of Data Processing Tools in Hadoop", Int. Conf. on Electrical, Electronics, Communication, Computer and Optimization Techniques, pp. 238-242, 2016, (DOI:10.1109/ICEECCOT.2016.7955222).
- [18] Surya Narayanan Swaminathan and Ramez Elmasri, "Quantitative Analysis of Scalable NoSQL Databases", IEEE International Congress on Big Data, pp. 323-326, 2016, (DOI: 10.1109/BigDataCongress.2016.49).
- [19] Dong-Her Shih, Feng-Chuan Huang, Wei-Hao Lai and Ming-Hung Shih, "Privacy Preserving and Performance Analysis on Not Only SQL Database and Aggregation in Bigdata Era", Int. Conf. on Cloud Computing and Big Data Analysis, pp. 56-60, 2017, (DOI: 10.1109/ICCCBDA.2017.7951884).
- [20] Ainhoa Azqueta-Alzuaz, Marta Patiño-Martinez, Ivan Brondino and Ricardo Jimenez-Peris, "Massive Data Load on Distributed Database Systems over HBase", IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 776-779, 2017, (DOI:10.1109/CCGRID.2017.124).