International Journal on Applications in Information and Communication Engineering
Volume 2: Issue 3: March 2016, pp 26-29. www.aetsjournal.com ISSN (Online) : 2394-6237

I^3 Map Reduce for mining Big Data through Density based Clustering

P.M. Anupraba , S.Parameswaran

Abstract— Everyday enormous amount of data is being produced worldwide. Big data is comprised of datasets too large to be handled by traditional database systems. Data mining is the technique that can discovers new patterns from large datasets, where most classical data mining methods became out of reach in practice to handle such big data. Big data is large volume, heterogeneous (STRUCTURED, SEMI STRUCTURED & UN STRUCTURED) distributed data, which can be analyzed by MapReduce, a substructure of HDFS Platform. In this paper, a new MapReduce model is proposed, which acts as an efficient approach for refreshing materialized data by reducing re-computation process. It is based on a new Density based clustering algorithm with its concepts of density reachability and density connectivity. Density based clustering algorithm has played a vital role in finding non linear shapes structure based on the density. I assessed the final results to show performance improvement.

IndexTerms **-** MapReduce, Density based Clustering, Big Data, Cloud

I. INTRODUCTION

Big Data became a big topic across nearly every area of IT. IDC defines Big Data technologies as a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data by enabling high-velocity capture, discovery, and/or analysis. There are three main characteristics of Big Data: the data itself, the analytics of the data, and the presentation of the results of the analytics. Then there are the products and services that can be wrapped around one or all of these Big Data elements. The digital universe itself, of course, comprises data all kinds of data.

However, the vast majority of new data being generated is unstructured. This means that more often than not, we know little about the data, unless it is somehow characterized or tagged a practice that results in metadata. Metadata is one of the fastest-growing sub segments of the digital universe (though metadata itself is a small part of the digital universe overall). We believe that by 2020, a third of the data in the digital universe (more than 13,000 hexa bytes) will have Big Data value, but only if it is tagged and analysed. Big data and analytics offer the promise to transform risk management and decision-making, providing more information and more speed. They won't solve every problem, however, and with these new sources of information come new pressures to focus risk management activities and respond quickly to perceived dangers. Using big data, companies have the potential to better identify "hidden" risk and allow better root cause analysis. Risk managers can improve their ability to determine the probability of an event by leveraging metadata and using customer segmentation to identify risk factors. Big data can help develop better early warning indicators that will allow companies to mitigate risk more effectively.

P.M. Anupraba , Pg Scholar M.E (Cse) , Sree Sowdambika College Of Engg, Aruppukottai, Tamil Nadu, India.
(Email : Anupraba.Pm@Gmail.Com)
S.Parameswaran Asst Prof, Dept Of Cse , Sree Sowdambika College Of Engg, Aruppukottai, Tamil Nadu, India.
(Email : Eswar232000@Gmail.Com)

II. MAP REDUCE BACKGROUND

A Map Reduce program is composed of a Map function and a Reduce function. The Map function takes a kvpair<K1;V1>as input and computes zero or more intermediate kvpairs<K2;V2>. Then all <K2;V2>are grouped byK2. The Reduce function takes a K2 and a list of {V2}as input and computes the final output kvpairs<K3; V3>.

MapReduce system runs a Job Tracker process on a master node to monitor the job progress, and a set of Task Tracker processes on worker nodes n to perform the actual Map and Reduce tasks.

The Job Tracker starts a Map task per data block, and typically assigns it to the Task Tracker on the machine that holds the corresponding data block inorder to minimize communication overhead. Each Map task calls the Map function for every input, and stores the intermediate kv-pairs on local disks. Intermediate results are shuffled to Reduce tasks according to a partition function (e.g., a hash function)on K2. After a Reduce task obtains and merges intermediate results from all Map Tasks and produces the final output results.

III. APACHE HIVE

Hive is a component of Hortonworks Data Platform (HDP). Hive provides a SQL-like interface to data stored in HDP. The Apache Hive ™ data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data. and query the data using a SQL-like language called HiveQL. At the same time this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

A. HBase:

Apache HBase is the Hadoop database modeled after Google's BigTable. It is a sparse, distributed, persistent multi-dimensional sorted map. The map is indexed by a row key, column key, and a timestamp. Each value in the map is an uninterpreted array of bytes.

B. Hive + HBaseIntegration :

For Input/OutputFormat, getSplits(), etc underlying HBase classes are used.

Column selection and certain filters can be pushed down. HBase tables can be used with other (Hadoop native) tables and SQL constructs. Hive DDL operations are converted to HBase DDL operations via the client hook. –All operations are performed by the client –No two phase commit.

C. Schema Mapping :

Hive table + columns + column types <=>HBase table + column families (+ column qualifiers). Every field in Hive table is mapped in order to either

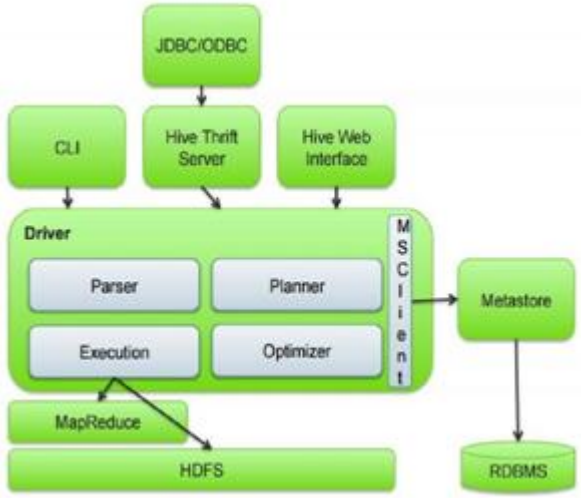
– The table key (using :key as selector)

– A column family (cf:) -> MAP fields in Hive

– A column (cf:cq) Hive table does not need to include all columns in HBase

D. Type Mapping :

Recently added to Hive Previously all types were being converted to strings in HBase. Hive has:Primitive types: INT, STRING,BINARY, DATE, etc –ARRAY –MAP –STRUCT. HBase does not have types–Bytes.toBytes() Table level property"hbase.table.default.storage.type" = "binary". Type mapping can be given per column after .



Apache HIVE Architecture

E. Bulk Load: Steps to bulk load:

– Sample source data for range partitioning

– Save sampling results to a file

– Run CLUSTER BY query using HiveHFileOutputFormat and TotalOrderPartitioner

– Import Hfiles into HBase table Ideal setup should be SET hive.hbase.bulk=true INSERT OVERWRITE TABLE web_table SELECT ….

F. Filter Pushdown :

Idea is to pass down filter expressions to the storage layer to minimize scanned data. To access indexes at HDFS or HBase.

G. Filter Decomposition :

Optimizer pushes down the predicates to the query plan. Storage handlers can negotiate with the Hive optimizer to decompose the filter .

mi;j (8i) and performs combine2(mi;j;vj) to obtain mvi;j. The second job groups the mvi;j and vi on the same i, performs the combineAll ({mvi;j}) operation, and updates vi using assign(vi;v 0 i).

MapReduce job per iteration, as shown in Algorithm 2.

Algorithm 1. PageRank in MapReduce

Map Phase input: $< i, N_i \mid R_i >$
1: output $< i, N_i >$
2: **for all** j in N_i **do**
3: $R_{i,j} = \frac{R_i}{|N_i|}$
4: output $< j, R_{i,j} >$
5: **end for**

Reduce Phase input: $< j, \{R_{i,j}, N_j\} >$
6: $R_j = d \sum_i R_{i,j} + (1-d)$
7: output $< j, N_j \mid R_j >$

Algorithm 2.GIMV in MapReduce

Map Phase1 input: $< (i,j), m_{i,j} >$ or $< j, v_j >$
1: **if** kv-pair is $< (i,j), m_{i,j} >$ **then**
2: output $< (i,j), m_{i,j} >$
3: **else if** kv-pair is $< j, v_j >$ **then**
4: **for all** i blocks in j's row **do**
5: output $< (i,j), v_j >$
6: **end for**
7: **end if**

Reduce Phase 1 input: $< (i,j), \{m_{i,j}, v_j\} >$
8: $mv_{i,j} = \text{combine2}(m_{i,j}, v_j)$
9: output $< i, mv_{i,j} >, < j, v_j >$

Map Phase 2: output all inputs

Reduce Phase 2 input: $< i, \{mv_{i,j}, v_i\} >$
10: $v_i' \leftarrow \text{combineAll}(\{mv_{i,j}\})$
11: $v_i \leftarrow \text{assign}(v_i, v_i')$
12: output $< i, v_i >$

IV. ITERATIVE COMPUTATION

Iteration in computing is the repetition of a block of statements within a computer program. It can be used both as a general term, synonymous with repetition, and to describe a specific form of repetition with a mutable state. Confusingly, it may also refer to any repetition stated using an explicit repetition structure, regardless of mutability. It is applied using some algorithms like PageRank, GIM-V,Clustering etc.

A. PageRank:

PageRank is an algorithm used by Google Search to rank websites in their search engine results.

PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google: PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.It computes a ranking score for each vertex in a graph. After initializing all ranking scores, the computation performs a

B. GIM-V:

Generalized Iterated MatrixVector multiplication (GIM-V) is an abstraction of many iterative graph mining operations (e.g., PageRank, spectral clustering, diameter estimation, connected components). These graph mining algorithms can be generally represented by operating on an nn matrix M and a vector v of size n. Suppose both the matrix and the vector are divided into sub-blocks. Let mi;j denote the ði;jÞth block of M and vj denote the jth block of v. The computation steps are similar to those of the matrix-vector multiplication and can be abstracted into three operations: (1) mvi;j = combine2(mi;j;v j); (2) v0 i = combineAlli({mvi;j}); and (3) vi = assign(vi;v 0 i). We can compare combine2 to the multiplication between mi;j and vj, and compare combineAll to the sum of mvi;j for row i. Algorithm 4 shows the MapReduce implementation with two jobs for each iteration. The first job assigns vector block vj to multiple matrix blocks

C. Density Based Clustering

Density based clustering algorithm has played a vital role in finding non linear shapes structure based on the density. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is most widely used density based algorithm. It uses the concept of density reachability and density connectivity.

D.Density Reachability

A point "p" is said to be density reachable from a point "q" if point "p" is within ε distance from point "q" and "q" has sufficient number of points in its neighbors which are within distance ε.

E. Density Connectivity

1)A point "p" and "q" are said to be density connected if there exist a point "r" which has sufficient number of points in its neighbors and both the points "p" and "q" are within the ε distance. This is chaining process. So, if "q" is neighbor of "r", "r" is neighbor of "s", "s" Start with an arbitrary starting point that has not been visited.

2)Extract the neighborhood of this point using ε (All points which are within the ε distance are neighborhood).

3)If there are sufficient neighborhood around this point then clustering process starts and point is marked as visited else this point is labeled as noise.

4)If a point is found to be a part of the cluster then its ε neighborhood is also the part of the cluster and the above procedure from step 2 is repeated for all ε neighborhood points. This is repeated until all points in the cluster is determined.

5)A new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

6)This process continues until all points are marked as visited.

F. Clusters obtained by DBSCAN

S.No	Eps	Min Point	No.of Clusters
1	0.5	3	8
2	1	3	8
3	0.5	4	5
4	1	4	5

V. CONFIGURATION

Install a stable release of Hive by downloading a tarball, or you can download the source code and build Hive from that.

A. Requirements

Java1.7 Hive versions 0.14 to 1.1 work with Java 1.6 as well. Users are strongly advised to start moving to Java 1.8

Hadoop 2.x (preferred), 1.x

Hive versions up to 0.13 also supported Hadoop 0.20.x, 0.23.x.

Hive is commonly used in production Linux and Windows environment. Mac is a commonly used development environment.

B. Configuration Management Overview

Hive by default gets its configuration from <installdir>/conf/hivedefault.xml. The location of the Hive

configuration directory can be changed by setting the HIVE_CONF_DIR environment variable.Configuration variables can be changed by (re-)defining them in <installdir>/conf/hive-site.xmlLog4j configuration is stored in <install-dir>/conf/hivelog4j.properties. Hive configuration is an overlay on top of Hadoop – it inherits the Hadoop configuration variables by default.Hive configuration can be manipulated by:Editing hive-site.xml and defining any desired variables (including Hadoop variables) in it .Using the set command.Invoking Hive (deprecated), Beeline or HiveServer2 using the syntax:

$ bin/hive --hiveconf x1=y1 --hiveconf x2=y2 //this sets the variables x1 and x2 to y1 and y2 respectively

$ bin/hiveserver2 --hiveconf x1=y1 -hiveconf x2=y2 //this sets server-side variables x1 and x2 to y1 and y2

respectively

$ bin/beeline --hiveconf x1=y1 --hiveconf x2=y2 //this sets client-side variables x1 and x2 to y1 and y2 respectively.

Setting the HIVE_OPTS environment variable to "--hiveconf x1=y1 --hiveconf x2=y2" which does the same as above.

Hive compiler generates map-reduce jobs for most queries. These jobs are then submittedto the Map-Reduce cluster indicated by the variable: mapred.job. tracker.

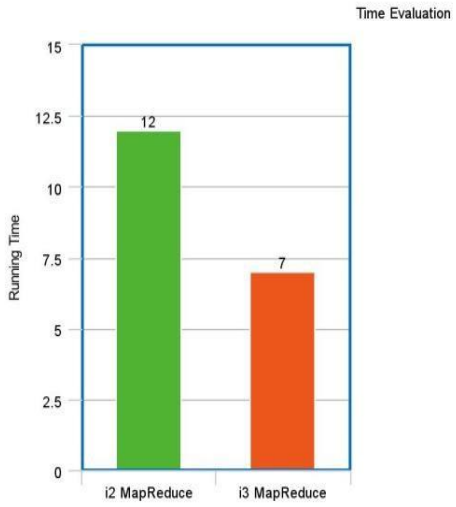
C.Hive Logging:

Hive uses log4j for logging. By default logs are not emitted to the console by the CLI.To configure a different log

location,set hive.log.dir in $HIVE_HOME /conf/hivelog4j. properties. Make sure the directory has the sticky bit set (chmod 1777 <*dir*>).

D.Runtime Configuration:

Hive queries are executed using map-reduce queries and, therefore, the behavior of such queries can be controlled by the Hadoop configuration variables. TheHiveCLI (deprecated) and Beeline command 'SET' can be used to set any Hadoop (or Hive) configuration variable.



VI. CONCLUSION & FUTURE WORK

In this paper, experience of designing and implementing i3 MapReduce isachieved by density based clustering. In Future, this work is continuing on the functions like,

(i) Define a sound methodology for selecting centroids
(ii) Fine tune some parameters such as the decrease of the membership as a function of cost
(iii) Improve the software implementation.
(iv) New method to compute fuzzy membership
(iv) Good results in difficult problems.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Commun. ACM, vol. 51, pp. 107113, 2008.
[2] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," presented at the Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, Lisbon, Portugal, 2007.
[3] Micheal Ankerst, M. M.-P. (1999). OPTICS: Ordering Points To Identify the Clustering Structure. Philadelphia: Proc. ACM SIGMOD'99 Int. Conf.
[4] Ester M., Kriegel H.-P., and Xu X. 1995. A Database Interface for Clustering in Large Spatial Databases, Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining, Montreal, Canada, 1995, AAAI Press, 1995. Garcfa J.A., Fdez-Valdivi
[5] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An ef- ficient data clustering method for very large databases. In Proceedings of SIGMOD, pages 103–114, June 1996.
[6] https://en.wikibooks.org/wiki/Data _Mining_Algorithms_In_R/Clustering/Density-Based_Clustering
[7] Density-based clustering algorithms – DBSCAN and SNN by Adriano Moreira, Maribel Y. Santos and Sofia Carneiro.
[8] Ester M., Kriegel H.-P., and Xu X. 1995. A Database Interface for Clustering in Large Spatial Databases, Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining, Montreal, Canada, 1995, AAAI Press, 1995. Garcfa J.A., Fdez-Valdivi