ISSN 2348 – 9928
IJAICT Volume 3, Issue 11, March 2017 Doi:01.0401/ijaict.2017.03.04 Published on 05 (04) 2017

REVOLUTION DATA ANALYTICS IN HADOOP

Ms. N. Rama Kalpana
Assistant Professor,
Department of Computer Science and Engineering,
Adithya Institute of Technology,
Coimbatore, TamilNadu, India

Abstract* – Big Data refers to datasets whose size are beyond the ability of typical database software tools to capture, store, manage and analyse.It is a new generation of technologies and architectures designed to extract value economically from very large volumes of awide variety of data by enabling high velocity capture, discovery and analysis. Big data is data that exceeds the processing capacity of conventional database systems[1]. The data istoo big, moves too fast, or does not fit the structures of existing database architectures. To gain value from these data, there must be an alternative way to process it.Bigdata analysis is used for analysis the huge number of data involved in traditional data processing. It includes analysis, capture, data curation, search, sharing, storage, transfer, visualization, querying and information privacy. Relational database management systems and desktop statistics and visualization packages have difficulty handling big data.

***Keywords* – Big Data, Relational Database, Information Privacy, Revolution Analytics.**

I. INTRODUCTION

Big Analytics delivers competitive advantage in two ways compared to the traditional analytical model. Big Analytics describes the efficient use of a simple model applied to volumes of data that would be too large for the traditional analytical environment. Research suggests that a simple algorithm with a large volume of data is more accurate than a sophisticated algorithm with little data.

The objectives for working with Big Data Analytics:

1. Avoid sampling / aggregation
2. Reduce data movement and replication
3. Bring the analytics as close as possible to the data
4. Optimize computation speed

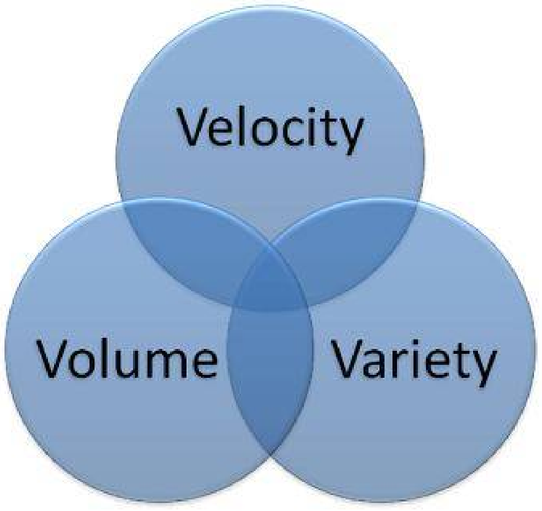
Revolution Analytics delivers optimized statistical algorithms for the three primary data management paradigms being employed to address growing size and increasing variety of organizations' data, including file-based, MapReduce (e.g. Hadoop) or In-Database Analytics. Open Source R was not built for Big Data Analytics because it is memory-bound. Depending on the type of statistical analysis required, Big Data also causes issues that is called "Big Computations," as some algorithms require a great deal of processing capacity on their own and may not lend themselves to running in every data management paradigm. Big Computations, parallelism (as we've deployed with IBM Netezza and ScaleR) is important to performance and to the accuracy of the statistical analysis. Coupled with an intuitive R Development Environment from Revolution Analytics, the degree of innovation exceeds that which may be achieved through packaged analytic applications.

II. CHARACTERISTICS OF BIG DATA

2.1 High Volume

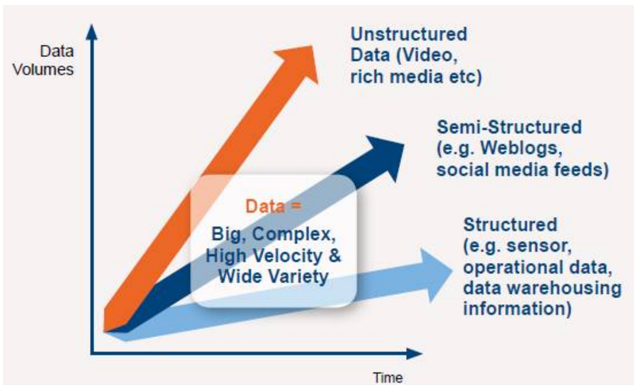
Big Data is not just about the size of data but also includes data variety and data velocity. Volume is synonymous with the "big" in the term, "Big Data".

Volume is a relative term – some smaller-sized organisations are likely to have mere gigabytes or terabytes of data storage as opposed to the petabytes or exabytes of data that big global enterprises have. Data volume will continue to grow, regardless of the organisation's size. There is a natural tendency for companies to store data of all sorts: financial data, medical data, environmental data and so on. Many of these companies' datasets are within the terabytes range today but, soon they could reach petabytes or even exabytes.



2.2 High Variety

Data can come from a variety of sources (typically both internal and external to an organisation) and in a variety of types. With the explosion of sensors, smart devices as well as social networking, data in an enterprise has become complex because it includes not only structured traditional relational data, but also semi-structured and unstructured data.


Data Volumes
Unstructured Data (Video, rich media etc)
Semi-Structured (e.g. Weblogs, social media feeds)
Structured (e.g. sensor, operational data, data warehousing information)
Data = Big, Complex, High Velocity & Wide Variety
Time

Structured data: This type describes data which is grouped into a relational scheme (e.g., rows and columns within a standard database). The data configuratio and consistency allows it to respond to simple queries to arrive at usable information, based on an organisation's parameters and operational needs.

Semi-structured data: This is a form of structured data that does not conform to an explicit and fixed schema. The data is inherently self-describing and contains tags or other markers to enforce hierarchies of records and fields within the data. Examples include weblogs and social media feeds.

Unstructured data: This type of data consists of formats which cannot easily be indexed into relational tables for analysis or querying. Examples include images, audio and video files.

2.3 High Velocity

Velocity of data in terms of the frequency of its generation and delivery is also a characteristic of big data. Conventional understanding of velocity typically considers how quickly the data arrives and is stored, and how quickly it can be retrieved. In the context of Big Data, velocity should also be applied to data in motion: the speed at which the data is flowing. The various information streams and the increase in sensor network deployment have led to a constant flow of data at a pace that has made it impossible for traditional systems to handle.

Informed intuition: predicting likely future occurrences and what course of actions is more likely to be successful.
Intelligence: looking at what is happening now in real time (or close to real time) and determining the action to take.
Insight: reviewing what has happened and determining the action to take.

III. REVOLUTION ANALYTICS AND HADOOP

Many enterprise companies are used to slove the big data analytics in "R" statistical programming language and Hadoop (both open source projects) as a potential solution for their organisations. Large amount of data especially unstructured data collected by organizations and enterprises explodes, Hadoop is emerging rapidly as one of the primary options for storing and performing operations on that data[2]. The marriage of R and Hadoop seems a natural one. Both are open source projects and both are data driven. But there are some fundamental challenges that need to be addressed in order to make the marriage work. Revolution Analytics is addressing these challenges with its Hadoop-based development.

3.1 Iterative vs. batch processing

In Iterative Process, explore and try to understand the data, try some different statistical techniques, drill down on various dimensions, etc. R is a powerful tool, and an ideal environment for performing such analysis. Hadoop on the other hand, is batch oriented where jobs are queued and then executed, and it may take minutes or hours to run these jobs.

3.2 In-memory vs. in parallel

R is designed to have all of its data in memory and programs in Hadoop (map/reduce) work independently and in parallel on individual data slices.

IV. REVOLUTION ANALYTICS CAPABILITIES FOR HADOOP

Revolution has created a series of "Revo Connect Rs for Hadoop" that will allow an R programmer to manipulate Hadoop data stores directly from HDFS and HBASE, and give R programmers the ability to write MapReduce jobs in R using Hadoop Streaming. RevoHDFS provides connectivity from tR to HDFS and RevoHBase provides connectivity from R to HBase. RevoHStream allows MapReduce jobs to be developed in R and executed as Hadoop Streaming jobs.

V. PHASES IN THE PROCESSING PIPELINE

5.1 Data Acquisition and Recording

It it is recorded from some data generating source

5.2 Information Extraction and Cleaning

Information collected will not be in a format ready for analysis.

5.3 Data Integration, Aggregation, and Representation

Heterogeneity of the flood of data, it is not enough merely to record it and throw it into a repository[3].

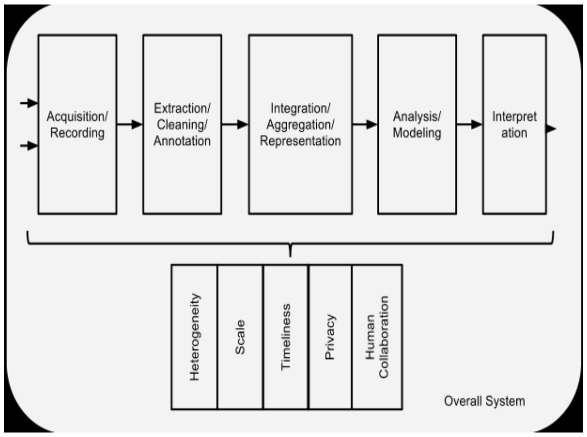
5.4 Query Processing, Data Modeling, and Analysis

Methods for querying and mining Big Data are fundamentally different from traditional statistical analysis on small samples

.

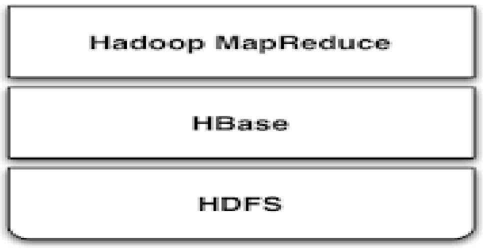
5.5 Interpretation

The ability to analyze Big Data is of limited value if users cannot understand the analysis.



VI. HADOOP DISTRIBUTED FILE SYSTEM

A basic storage mechanism in Hadoop is HDFS (Hadoop Distributed File System). For an R programmer, being able to read/write files in HDFS from a standalone R Session is the first step in working within the Hadoop ecosystem. The memory constraints of R, this capability allows the analyst to easily work with a data subset and begin some ad hoc analysis without involving outside parties .It also enables the R programmer to store models or other R objects that can then later be recalled and used in MapReduce jobs. When MapReduce jobs finish executing, they normally write their results to HDFS. Inspection of those results and usage for further analysis in R make this functionality essential.

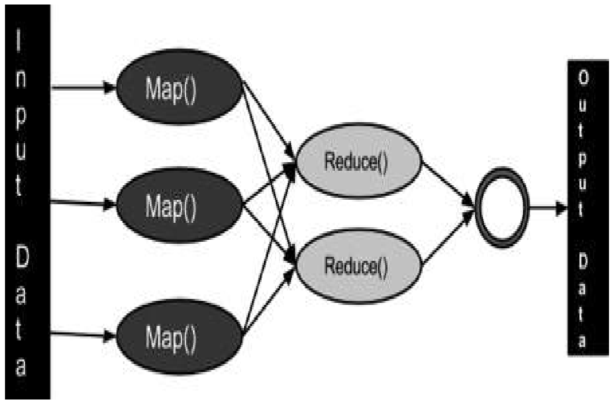


6.1 HBASE Overview

HBASE is the top layer in the HDFS. In HBASE, Hadoop's answer to providing database likes table structures. Just like being able to work with HDFS from inside R, access to HBASE helps open up the Hadoop framework to the R programmer. It is not be able to load a billion-row- by-million-column table, working with smaller subsets to perform ad hoc analysis can help lead to solutions that work with the entire data set.

6.2 MapReduce – Data Reduction

The processing pillar in the Hadoop ecosystem is the MapReduce framework. The framework allows the specification of an operation to be applied to a huge data set, divide the problem and data, and run it in parallel. A very large dataset can be reduced into a smaller subset where analytics can be applied[2]. In a traditional data warehousing scenario, this might entail applying an ETL operation on the data to produce something usable by the analyst. In Hadoop, these kinds of operations are written as MapReduce jobs in Java. There are a number of higher level languages like Hive and Pig that make writing these programs easier. The outputs of these jobs can be written back to either HDFS/HBASE or placed in a traditional data warehouse. R can then be used to do the analysis on the data.



6.3 MapReduce – R

Executing R code in the context of a MapReduce job elevates the kinds and size of analytics that can be applied to huge datasets. It involves pushing the model to the Task nodes in the Hadoop cluster, running a MapReduce job that loads the model into R on a task node, scoring data either row-by row (or in aggregates), and writing the results back to HDFS. In the most simplistic case this can be done with just a Map task.

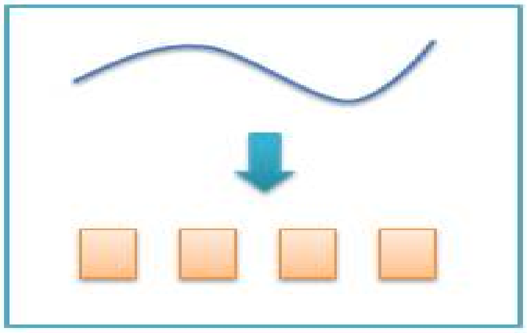
Visualizations of huge datasets can provide important insights that help understand the data. Creating a binning algorithm in R that is executed as a MapReduce job can produce an output that can be fed back into an R client to render such visualizations. It include data Mining algorithms like K-Means clustering, logistic regression with small numbers of parameters and iterations, and even linear regression.

6.4 MapReduce – Hybrid

A hybrid model that combines using something like HIVE QL, and R. HIVE QL allows us to perform some SQL like capabilities to create naturally occurring groups where R models can be created. Revolution has created an R package that allows creation of MapReduce jobs in R. The goal is providing a simple and usable interface that allows specification of both Map and Reduce as functions in R. It keeps the data scientist working in R. R programmer might have to rethink the approach to how algorithms can be realized and implemented. Revolution Analytics is the leading commercial provider of software and services based on the open source R project for statistical computing.

VII. BINNING ALGORITHM

Data binning or bucketing is a data pre-processing technique used to reduce the effects of minor observation errors. Statistical data binning is a way to group a number of more or less continuous values into a smaller number of "bins". Binning or discretization is the process of transforming numerical variables into categorical counterparts.



Two types of binning algorithms:

1. Unsupervised Binning
2. Supervised Binning.

Parameters : x, y : array
The x and y data values.
yerr : array, optional
Errors on the data values.

x0 : float, optional
Starting time of first bin. Default is lowest given x value.
dt : float, optional
Width of a bin (either dt, nbins or reduceBy must be given).
nbins : int, optional
Number of bins to use (either dt, nbins or reduceBy must be given). Note that this specifies the number of bins into which the range from x0 to the last data point is subdivided.

reduceBy : int, optional

Reduce the number of elements in the array by the given factor (either dt, nbins or reduceBy must be given). Note that in this case, x0 is set to the first (minimum x-value) and the number of bins, n, is calculated according to the prescription: n = int(round(len(x)/reduceBy))

removeEmpty : boolean, optional

If True (default), bins with no data points will be removed from the result.

removeNoError : boolean, optional

If True, bins for which no error can be determined will be removed from the result. Default is False.
useBinCenter : boolean, optional

If True (default), the time axis will refer to the center of the bins. Otherwise the numbers refer to the start of the bins.

useMeanX : boolean, optional
If True, the binned x-values refer to the mean x-value of all points in that bin. Therefore, the new time axis does not have to be equidistant.

nanHandling : None, “ignore”, float, (optional)
Controls how NaNs in the data are handled.
None: By default (None), nothing is done and NaNs are treated as if they were valid input data, so that they are carried over into the binned data. This means that output bins containing NaN(s) will also end up as NaN(s). If ‘ignore’

‘ignore’: In this case, NaNs contained in the input data are removed from the data prior binning. Note however, that x0, unless specified explicitly, will still refer to the first data point, whether or not this holds a NaN value.
float: If a float is given, input data values containing NaNs are replaced by the given float before binning. Note that no error on the data (yerr) can be considered in this case, to avoid erronous treatment of un- or misspecified error values.

Returns :
Binned data set : array
An array with four columns: 1) The new x-axis, 2) The binned data (the mean value of the data points located in the individual bins), 3) Error of binned data, 4) The number of input data points used to create the bin. For instance, the new x-values can be accessed using result[::,0].

dt : float
The width of the bins.

VIII. K MEANS CLUSTERING ALGORITHM

It is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids shoud be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done.

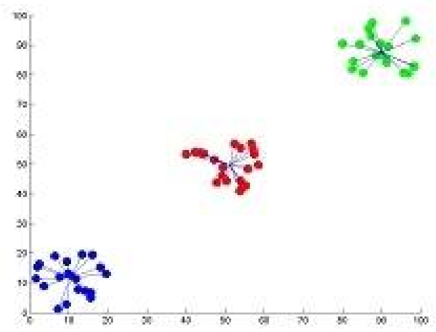
$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} (\|x_i - v_j\|)^2$$

The objective function
where,
'||*xi - vj*||' is the Euclidean distance between *xi* and *vj*.
'*ci*' is the number of data points in i^{th} cluster.
'*c*' is the number of cluster centers.

Algorithmic steps for k-means clustering
Let X = {x1,x2,x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} be the set of centers.
1) Randomly select '*c*' cluster centers.
2) Calculate the distance between each data point and cluster centers.
3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..
4) Recalculate the new cluster center using:

$$v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$$

where, '*ci*' represents the number of data points in i^{th} cluster.
5) Recalculate the distance between each data point and new obtained cluster centers.
6) If no data point was reassigned then stop, otherwise repeat from step 3).
Advantages
1) Fast, robust and easier to understand.
2) Relatively efficient: O(tknd), where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d << n.
3) Gives best result when data set are distinct or well separated from each other.



IX. CONCLUSION

Better analysis of the large volumes of data that are becoming available, there is the potential for making faster advances in many scientific disciplines and improving the profitability and success of many enterprises.The challenges include not just the obvious issues of scale, but also heterogeneity, lack of structure, error-handling, privacy, timeliness, provenance, and visualization, at all stages of the analysis pipeline from data acquisition to result interpretation. These technical challenges are common across a large variety of application domains, and therefore not cost-effective to address in the context of one domain alone.

References

[1] Edd Dumbill. What is big data? [Online] Available from: http://radar.oreilly.com/2012/01/what-is-big-data.htm l[Accessed 9th July 2012].
[2] http://www.revolutionanalytics.com/big-analytics.
[3] Materials Genome Initiative for Global Competitiveness. National Science and Technology Council. June 2011.
[4] The k-means algorithm - Notes by Tan, Steinbach, Kumar Ghosh.
[5] http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html