International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 9 (2015) pp. 22133-22144

http://www.ripublication.com

Fast Approximate Aggregation Algorithms For Effective Browser Visualization Using Similarity Heuristics

Shiju Sathyadevan[1], Hariram S[2], Akhil Antony[3], Mahith VP[4]
[1]*Assistant Professor, Amrita Center for cyber security, Amrita University, Kerala, India*
[2]*Research Associate, Amrita Center for cyber security, Amrita University, Kerala, India*
[3]*PG Student, Department of Computer Application, Amrita University, Kerala, India*
[4]*PG Student, Department of Computer Application, Amrita University, Kerala, India*

Abstract

Real world data is processed in multiple servers which constitutes a huge collection of data that causes the browser to crash at regular interval of time. The Aggregation have been done to those data that are generated from various sources like log files, video, text, image etc .This paper combines the fusion of hierarchical clustering with Map-reduce which indicates the grouping of same type of data and reducing it in to key-value pairs ,this classification makes the processing and retrieval of real time data easy. Visualization of these data increases the efficiency for user to analyze it .The research results provide a better way to aggregate huge amount of data with high efficiency.

Key Words: Map Reduce, Hierarchical clustering, MRGF algorithm, Grain algorithm, Visualization

Introduction

Big data is one of the major challenges faced by industry today .With tremendous amount of data flowing into the organizations today, traditional architectures are not up to the demand. Clients are getting dissatisfied due to the under performance of the browser in handling large volume of data , which causes the browser to crash after a certain limit . Since none of the browser is able to handle big data and now a day's more users are expecting self-service access to the information from browser in a form that they can easily understand and share with others. This raises the questions like how do you present these huge chunks of data to the browser and to present it to the user in a understandable manner . This is not a minor consideration. Mining of data creates a big headache for analysts related to categorizing and presenting of data.

The effective way for handling data in a browser is a major challenge for achieving system performance as well as user satisfaction.

For this consider pairing visual analytics with big data through aggregation and there by representing these aggregated data in the browser. In today's technology what is important is the speed and agility, companies not only have to find and inspect the complete data they also need to find it quickly. Visualization helps organizations perform analyses and make decisions much more quickly [6], but the challenge increases as we go through bulkier volumes of data and accessing it at a immense speed. Some vendors are using heightened memory resources and powerful parallel processing to crunch larger volumes of data extremely quick. It takes a lot of compassion to get data in a right shape so that you can use visualization as part of data analysis and to get this done, we must first aggregate it using hierarchical clustering with Map-reduce[15]. Here one of the major challenges is the graphical representation of the stored data in a meaningful way , previously various techniques have been used like bar graphs, textual representation etc but all this model fails when their huge terabytes of data that have to be represent considering the fact of accuracy and tangibility. So it's time to move ahead with technologies like Hexagonal Binning, Circle Packing, Zoomable Circle Packing, heat map, Venn diagrams etc . These types of graphical representations of data made possible by visualization can be used to extract knowledge much faster than tables containing huge numbers and text[4].

In this paper, we review the theoretical foundations of the mining techniques and the algorithms are critically reviewed in a step by step manner along with a schematic representation of our algorithm. Finally we highlight the various techniques which we have used in our implementation for displaying the aggregated data in the browser. These research issues should be addressed in order to visualize robust systems that are capable of fulfilling the needs of the current technology.

The paper is framed as follows. Section 2 presents the details about the theoretical background for which we adopt two algorithms from different clustering classes to highlight the wide-range applicability, In Section 3 our proposed algorithms are transcribed, Section 4 describes the implementation of the Map-reduce interface fetched towards our cluster-based computing environment, Execution results of our proposed system have been explained in Section 5, Finally section 6 concludes this review paper.

Theoretical Background

This section manifest the efficiency of clustering algorithms, here we adopt two efficient clustering algorithms k-means [5][9] and Greedy agglomerative clustering [8]. While k-means is a representative of the clustering partitioning algorithm, Greedy agglomerative clustering is from a completely different class named Hierarchical Clustering.

Before going in the depth of K-Means and Greedy agglomerative clustering we have to know about the details of canopy clustering .Canopy clustering supports to scale clustering techniques [11]. It is often considered as per-processing step ahead of the Hierarchical agglomerative clustering algorithm or the K-means algorithm. Both

of these algorithms are used to enhance the efficiency of clustering applications on bulky data sets .Using canopy clustering the first main task is to portioning the overlapping canopies using distance metric then these data have been clustered using widely used clustering algorithms like Greedy agglomerative clustering and K-Means .But K-means algorithm are usually more costly due to more expensive distance measures. K-means comes under the class of partitioning clustering algorithm that partitions n objects into some favoured k number of clusters, where k is always less than n. Then we calculate the mean point and the algorithm is continuously repeated by combining each data to the nearest centroid and thereby discovering a new set center. This process is continued until assemblage, which is determined by observing that the centroids will no longer change upon successive iterations. It may not guarantee a universally ideal solution and can in fact merge to a wrong answer and one more alternative disadvantage of k-means is the inappropriate input value of k may yield into poor results.

By using the concept of canopy clustering allows these expensive distance comparisons to be made only between members of a particular canopy rather than all the members in the data set. Canopy clustering can also be treated as a divide-and-conquer heuristic in the sense that it grands an optimal solution to be reached expeditiously. However, the divide and- conquer strategy of canopy clustering may be essential in very large data sets where the space and time complexity are of major concern and this can be archived through integrating the concept of canopy along with greedy and map reduce frame work.

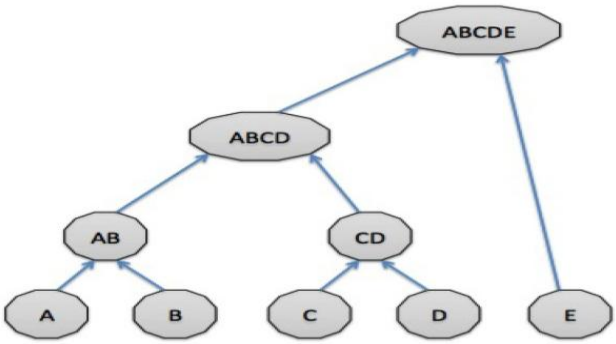

Figure 1: Agglomerative Hierarchical Clustering

Greedy Agglomerative Clustering is one of the most acceptable clustering techniques used to group items together based upon their similarity. In a standard greedy agglomerative clustering, we are given an input a set of items and a means of computing the distance between them. Then the two closest clusters are made to combine together until the clusters have been reduced to a target numbers shown in fig 1 . We use a standard implementation of greedy agglomerative clustering. Here first itself we have to initialize each element to be a cluster of size one, then compute the distances between all pairs of clusters, sort the distances from smallest to largest, and then repeatedly merge the two clusters which are closest together until one is left

with the desired number of clusters. A Greedy Agglomerative Cluster can drastically reduce this required number of estimations. We are guaranteed that any two points that will not fall into the same cluster we need not have to calculate the distances between these pairs of points. This enormously diminishes the required number of distance calculations for greedy agglomerative clustering and now combining this concept with map-reduce framework makes the job even easier. Map-reduce makes use of the JavaScript functions to implement the map and reduce operations. It will split into independent subtasks that can be processed in parallel [14]. First itself we must specify a mapping function as well as a reducing function to processes the data for one particular key at a time. Each step takes the input from the output of the previous step of Map-Reduce operation. Users must first assign a map function that processes an initial key/value pair , and then a reduce function which mainly concentrates on merging all the associated values linked with the initial key/value pairs of the map function as shown in fig 2.

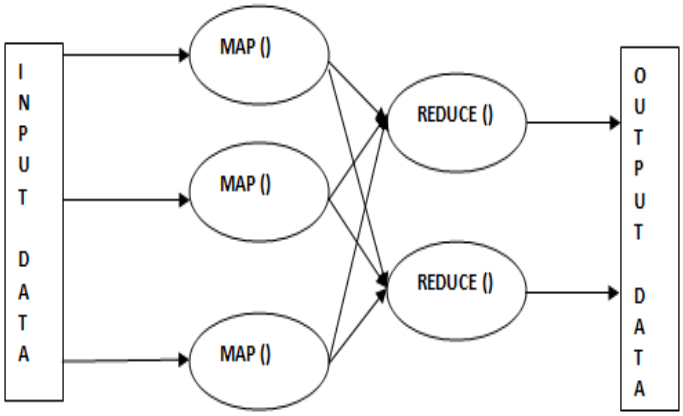

Figure 2: Map Reduce

Transcription of Our Algorithm

In this section we illustrate the algorithms which we have implemented and there are basically two algorithms MRGF (Map-Reduce Greedy agglomerative fusion algorithm) and Grain algorithm using probability

Schematic Representation MRGF

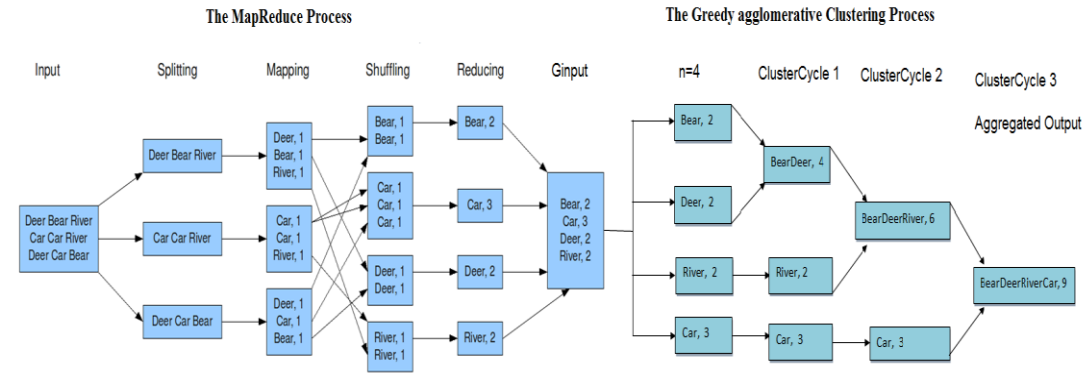

Figure 3:

MRGF Algorithm

Step 1: INPUT: Total number of elements N with at-least N/4 non-distinct elements in an Array
Step 2: Read the data from the user (key).
Step 3: Split the array in to multiple arrays. The probability of the partition index is proportional to .number of . elements N.
Step 4: Generate value to each key by using a counter variable
Step 5: The string value will be assigned as the key part of the < key, value > pair.
Step 6: Shuffle the data for grouping similar Keys
Step 7: Send the < key, value > pair to reducer.
Step 8: {On each reducer}
Step 9: Collect the < key, value > pair records fed by the mapper.
Step 10: Perform Local clustering on data. Due to the Map Reduce design, each individual reduce task works on the records that have been sorted based upon the key value. Therefore, the local clustering algorithm is indeed applied within each partition.
Step 11: Generate each key value by adding the value of each keys from each partitions
Step 12: {Greedy agglomerative clustering} n:no.of keys
Step 13: Read the reduced key-value pairs
Step 14: Order all the keys according to its values
Step 15: Find out the cluster cycles for aggregation. It should be always n-1
Step 16: Find sum of values in each cluster cycle (using single linkage formula)
Step 17: The final cluster cycle produce Combine all keys in to one key and add all values of each key
Step 18: Treat this new value as a new dimension for drill down
Step 19: Visualize the data on a predefined chart

Grain Algorithm Using Probability

Step 1: INPUT: Take the value key from the reducer output and the final key value obtained in MRGF algorithm.
Step 2: Fetch each cluster item ie a[key][value]
Step 3: Begin for(i=0;i<a.length();i++)
int p(a[i][0]) = a[i][1] / final key value
end for
Step 4:if user select any item i
int V=final key value * p(a[i][0])
end if
Step 5: Repeat i th key V number of times in visualization

Implementation

Many distinct implementations technique of MapReduce are possible. The right choice depends on the right environment which we have taken to implement our idea . Here we propose a novel programming approach using well-defined technique named data aggregation among all the mining techniques explained [2].Aggregation is the process which has been archived through the fusion between map reduce and greedy agglomerative clustering. Through fusion our main aim is to aggregate the data up to the maximum level. So that all the information's are gathered and expressed in a encapsulated form. The major purpose of aggregation is to collect more information's about particular group based on distinct variables. This process is more useful in a distributed environment. The most popular distributed concept is based on map-reduce framework because of its simplicity [15]. Map reduce is the framework which we have used to reduce the duplicated data and then these data should be made into a key-value format. Based on this concept the algorithm is mainly divided in 3 main steps: map, reduce and greedy clustering. The Map function is written by the user and its made to collect the input from the user and produces a set of key/value pairs and then its forwarded to the Reduce function and in greedy clustering the estimation is based upon a set of input key/value pairs from the reduce function, and produces a set of output key/value pairs which is in an aggregated format for browser visualization of data **.** Here we starts with a mapping function which consists of a group of random values and ends in a pair of aggregated values for visualization as demonstrated in fig 4 which shows the overall flow of our implementation**.**

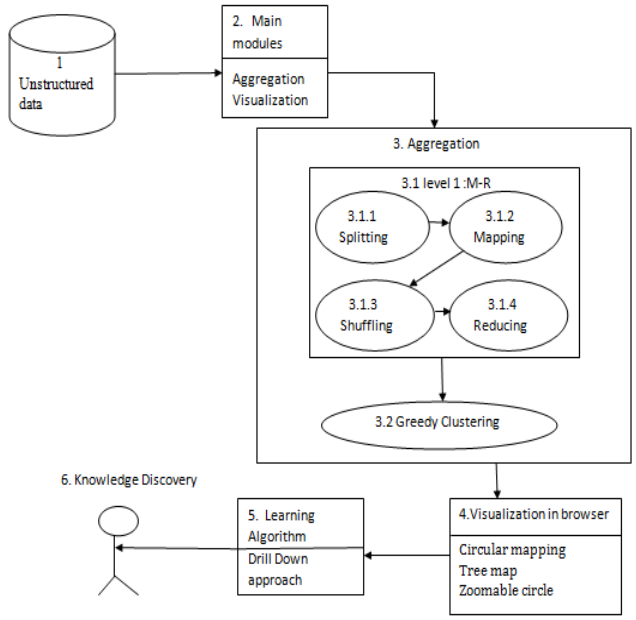

Figure 4:

The numbers tagged in fig 4 resembles to the numbers in the list-below

1) Load unstructured data
The unstructured data will be in typical JSON format which will be upload by the user.

2) The Main modules
Here there will be basically two main modules Aggregation and visualization .So at first the unstructured data will be loaded to the aggregation part for making the data in a structured format.

3) Levels of Aggregation
The data that have been revived from the main module is then aggregated using the concept of MapReduce which mainly consists of various sub topics like:

Spiting the data
Here the unstructured data have been divided into small chunks and further on the operation will be performed in this chunks of data.

Mapping similar data
Those data that have been acknowledged from spiting phase are then mapped into groups based on its similarity .Each map task reads from the input that is distributted to it. It parses the data and generates (key, value) pairs for data of interest.

Shuffle the data
The data that have been reserved from the map function are then redistributed based upon the key value of the map function , such that all data that belongs to the same key is located at the same place.

Reduce data
With data sorted by keys, the user calls the Reduce function. The Reduce function is being called once for each unique key. Each unique key represents a unique value that have to be displayed and the duplicated data have been bypassed here.

Greedy Agglomerative
Here we mainly focus on the whole map reduce data and convert the whole data into a single key value pair .

4) Visualization
Our next step is to visualize the aggregated data on to the browser using one of the various visualization techniques like Hexagonal Binning, Circle Packing, Zoomable Circle packing, heat map, venn diagrams[4] etc .These aggregated data can be comfortably handled by Visualization techniques and the aggregated result are represented on top layer of the any visualization techniques in a generalized form as shown in fig-6.

5) Drill down approach
Here the user will have an inner view of the aggregated data that have been obtained from the previous step .Here basically there will be a background algorithm which will perform the fetching of data that the user wants . So in short here we perform a drill down approach for fetching the original data from the output obtained in visualization .

6) Knowledge discovery
The original data that have been obtained from the drill down approach can be used to extract the knowledge and this concludes the flow of our implementation.

Execution Result

In this section we describes our experimental result of our implementation, the entire flow of our implementation have been explained here.

The situation of an existing system when the unstructured data have been loaded in the browser for visualization have been shown in fig 5

Limitations of Existing System

- In existing system user can’t segregate the data.
- Browser can’t handle huge volume of data

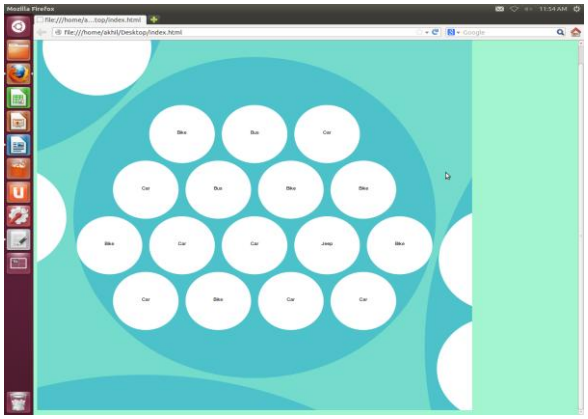

Figure 5:

In our shows our proposed system we first aggregate the data based on our MRGF algorithm in fig 6 and grain algorithm in fig6.1,fig6.2,fig6.3

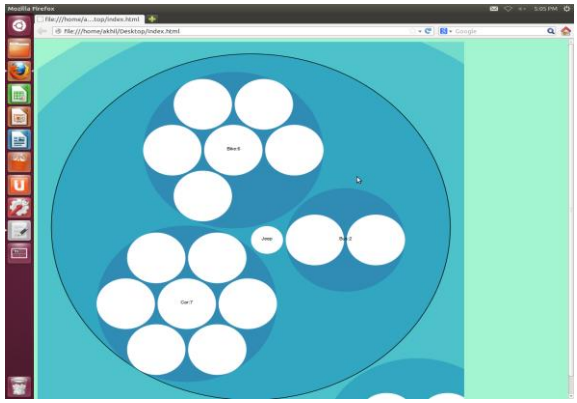

Figure 6:

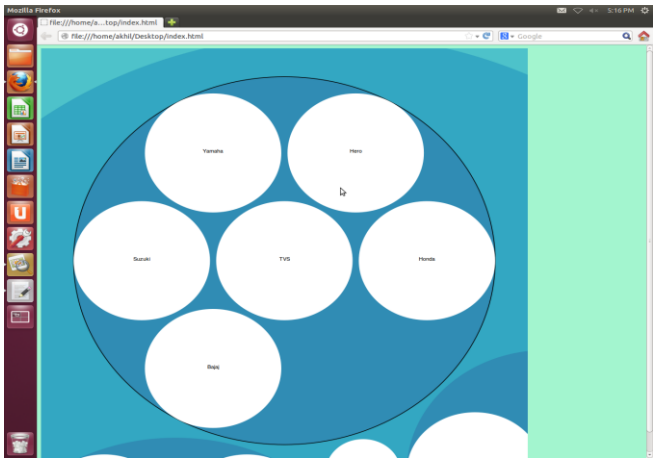

Figure 6.1:

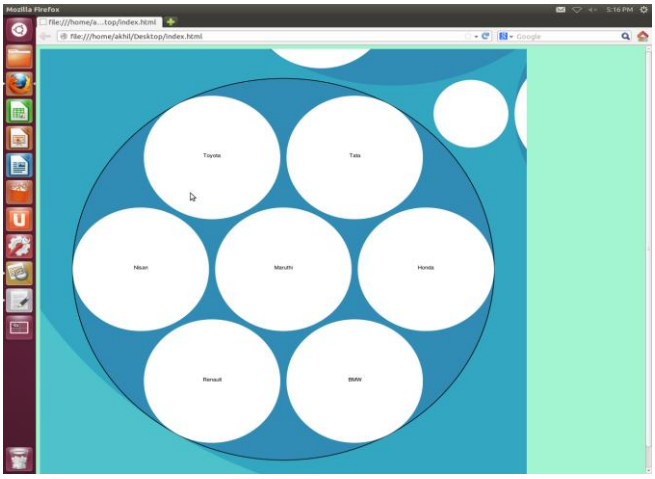

Figure 6.2:

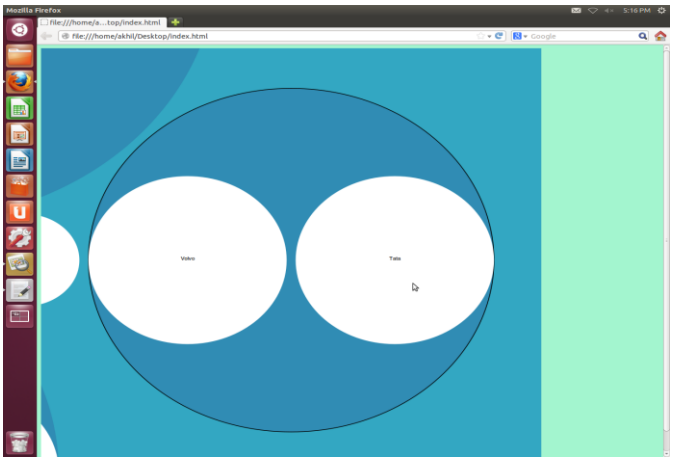

Figure 6.3:

From our proposed system user can easy segregate the data and extract the knowledge.

Conclusions

The experiments performed this review paper have shown that the Map Reduce model is a suitable alternative to support large-scale clustering of data for browser visualization. Analysis show the scalability, performance and accuracy of our method in processing mass data for visualization. The contributions of our work lie in both design and in the implementation of our MRGF and Grain algorithm. We use speedup and size up mechanism to evaluate the performances of our proposed algorithm. The results show that the our proposed algorithm can process large datasets effectively.

In the future it is feasible that the demand for cluster computing will increase as problem sets becomes bulkier and gather more interest developments in this field.

Map Reduce will definitely become a more favored solution and much used paradigm using hadoop or spark[16] because both of them are open-source framework and provides exceptional opportunities for analysts to handle real-world problems in a distributed system and this data can be made to visualize using techniques like heat map[3].

Acknowledgement

We thank Mr. Hariram S for helpful discussions and points for our relevant work. We thank Mr. Shiju Sathyadevan for advice on the presentation of this paper.

References

[1] Albert Bifet "Mining Big Data in Real Time.", International Journal of Informatica 15–20, December 15 2012.

[2] Mohamed Medhat Gaber, Arkady Zaslavsky and Shonali Krishnaswamy Research paper on" Mining Data Streams: A Review" SIGMOD Record, Vol. 34, No. 2, June 2005

[3] Johannes Trame and Carsten Keßler" Exploring the Lineage of Volunteered Geographic Information with Heat Maps "

[4] Zhicheng Liu, Biye Jiangz and Jeffrey Heer" imMens: Real-time Visual Querying of Big Data" Volume 32 (2013), Number 3

[5] Ran Jin, Chunhai Kou, Ruijuan Liu and Yefeng Li "Efficient parallel spectral clustering algorithm design for large data sets under cloud computing environment"2013 Page 1-10

[6] Chia Yuan Chuang, Ross Maciejewski, Student Member, IEEE, Stephen Rudolph, Ryan Hafen, Ahmad M. Abusalah, Mohamed Yakout, Mourad Ouzzani, William S. Cleveland, Shaun J. Grannis, and David S. Ebert, Fellow, IEEE" A Visual Analytics Approach to Understanding Spatiotemporal Hotspots". Paper 4 Review

[7] United Nations Global Pulse, http://www.unglobalpulse.org.

[8] K.Sasirekha and P.Baby "Agglomerative Hierarchical Clustering Algorithm-A Review" International Journal of Scientific andResearch Publications, Volume 3, Issue 3, March 2013 1 ISSN 2250-3153

[9] Guillaume Cleuziou "Applying an extended version of the k-means method for overlapping clustering",1999.

[10] Arindam Banerjee and Chase Krumpelman. "Model-based Overlapping Clustering ",1998

[11] Andrew McCallum,Kamal Nigam,Lyle H. Ungar "Efficient Clustering of HighDimensional Data Sets with Application to Reference Matching"2001.

[12] Guillaume Cleuziou and Lionel Martin and Christel Vrain. "PoBOC: an Overlapping Clustering Algorithm",1997.

[13] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, Christos Kozyrakis_" Evaluating MapReduce for Multi-core and Multiprocessor Systems
[14] Jeffrey Dean and Sanjay Ghemawat " MapReduce: Simplified Data Processing on Large Clusters"2004
[15] Apache Hadoop, http://hadoop.apache.org