UGC Approved Journal

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 9, September 2017

A Robust Density - Based Clustering Approach using DBCURE - MapReduce Techniques

Dhivya. N[1], Dr. G. Shrivakshan[2]

Research Scholar, P.G. and Research Department of Computer Science, Swami Vivekananda Arts and Science College, Villupuram, Tamilnadu, India[1]

Head and Assistant Professor, P.G. and Research Department of Computer Science, Swami Vivekananda Arts and Science College, Villupuram, Tamilnadu, India[2]

Abstract: Clustering is a most popular data mining technique. It is designed to discover an inherent natural structure of the data items, where objects in the same cluster are as similar as possible and data items in different clusters are as dissimilar. The DBSCAN and OPTICS are widely used clustering algorithms in density-based clustering. As there is a challenging problem in clustering that is because of an increasing trend of applications to deal with large volume of data. So that recently parallelizing clustering algorithms on large cluster of commodity machines using the MapReduce framework have received a lot of attention. In this paper, we propose DBCURE a novel density based clustering algorithm. It is a robust algorithm to discover the varying densities and is conveyable to parallelize with MapReduce. Concerning to tradition the density-based algorithms find clusters in a serial order. But in this proposed DBCURE-MR finds multiple clusters in a parallel approach. This work prove that DBCURE and DBCURE-MR find the clusters in a correct manner based on the definition of density-based clustering. The experimental results with different kinds of data sets prove that DBCURE-MR finds clusters efficiently without any deviation in finding clusters of varying densities and well balancing with the MapReduce framework.

Keywords: MapReduce, DBCURE, Density based Clustering, Parallelization algorithm.

I. INTRODUCTION

Data mining is the exploration and analysis of large data sets, in order to discover meaningful pattern and rules. The key idea is to find effective way to combine the computer's power to process the data with the human eye's ability to detect patterns. The objective of data mining is to design and work efficiently with large data sets. Data mining is the component of wider process called knowledge discovery from database. [4]. Data Mining is the process of analyzing data from different perspectives and summarizing the results as useful information. It has been defined as "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". The definition of data mining is closely related to another commonly used term knowledge discovery [2].Data mining is an interdisciplinary, integrated database, artificial intelligence, machine learning, statistics, etc. Many areas of theory and technology in current era are databases, artificial intelligence, data mining and statistics is a study of three strong large technology pillars. Data mining is a multi-step process, requires accessing and preparing data for a mining the data, data mining algorithm, analysing results and taking appropriate action. The data, which is accessed can be stored in one or more operational databases. In data mining the data can be mined by passing various process.Data mining consists of extract, transform, and load transaction data onto the data warehouse system, store and manage the data in a multidimensional database system, by using application software analyze the data, provide data access to business analysts and information technology professionals, present the data in a useful format, like a graph or table. Data mining involves the anomaly detection, association, classification, regression, rule learning, summarization and clustering.

II. LITERATURE REVIEW

Dr. E. Kesavulu Reddy et. al [10],explains thevarious Clustering technique that is used for finding hidden patterns in data mining. The different clustering techniques are stated Hierarchical clustering, Partitioning clustering, Density-based clustering and Grid-based clustering. The advantages of the paper is to presents the overview of the algorithms used in different clustering techniques along with their respective advantages and disadvantages. The different clustering methods are focused i.e. partitioning clustering, hierarchical clustering, density based clustering and grid based clustering. Under partitioning method, a brief description of k-means and k-Medoids algorithms have been

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2017

studied. In hierarchical clustering, the BIRCH and CHAMELEON algorithms have been described. The DBSCAN [9] and DENCLUE algorithms under the density based methods have been studied. Finally, under grid-based clustering method the STING and CLIQUE algorithms have been described. The comparisons with clustering analysis are mainly that different clustering techniques give substantially different results on the same data.

Jeff Dean et. al[18] , describes the framework definition and work process of MapReduce technique. It defines that MapReduce provides automatic parallelization and distribution, fault tolerance, I/O scheduling, monitoring and status updates. The advantage of this framework is to detect the failure via periodic heartbeats and re-execute in progress reduce tasks. The disadvantage of this framework is slow workers and delay in completion time.

Richard Bellman et. al [15], stated that the basic idea of dynamic programming is that of viewing an optimal policy as one determining the decision required at each time in terms of the current state of the system. An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions. The advantage of this work is a number of mathematical problems occurring in connection with the control of engineering economic systems are solved using this solution.

Prof. R. A. Fadnavis et. al [17],explains the Hadoop Distributed File System (HDFS) is designed to store large data sets reliably and to stream those data sets at high bandwidth to user applications. Hadoop is a popular open source implementation of MapReduce for the analysis of large datasets. To manage storage resources across the cluster, Hadoop uses a distributed user-level file system. This paper analyzes the performance of two major clustering algorithms K-means and DBSCAN on Hadoop platform and uncovers several performance issues. The merits of this framework is toprovide both reliability and data motion to applications. The main disadvantage is deficiency with K-means clustering algorithm is we have to set the number of clusters to be generated in advance and it is also sensitive to noise and outlier data. As per the results obtained K-means clustering algorithm performed on Hadoop alleviates the problem of time delay caused by increasing the number of nodes.

Lei Gu, Bo Li et. al [22],proposed an improved DBSCAN Clustering Method Based on Map Reduce and Grid presents that G-DBSCAN algorithm is developed from DBSCAN. G-DBSCAN algorithm reduces the number of query object as a starting point. Put the data into the grid, with the center point of the data in the grid to replace all the grid points as the algorithm input. The query object will be drastically reduced, thus improving the efficiency of the algorithm, reduces the memory footprint. In order to make the G-DBSCAN algorithm can adapt to large data processing, we will parallelize the G-DBSCAN algorithm, and combining it with Map Reduce framework. The results prove that G-DBSCAN and MRG-DBSCAN algorithm are feasible and effective. The advantage of this work is to removes the noise point and also reduces the memory footprint. The disadvantage of this paper to increasing scale of spatial database, data information becomes more and more complex. So in many applications it is difficult to select the appropriate clustering parameters. Therefore, the development of adaptive clustering algorithm will become an important part of our future research.

M. Ankerst et. al [20], defines an ordering based clustering that is OPTICS algorithm creates an ordering of a database, additionally storing the core-distance and a suitable reachability-distance for each object. We will see that this information is sufficient to extract all density-based clustering with respect to any distance € which is smaller than the generating distance e from this order. The main advantage of this approach, when compared to the clustering algorithms proposed in the literature, is that we do not limit ourselves to one global parameter setting. Instead, the augmented cluster-ordering contains information which is equivalent to the density based clustering corresponding to a broad range of parameter settings and thus is a versatile basis for both automatic and interactive cluster analysis. The disadvantage is infeasible to apply it in its current form to a database containing several million high-dimensional objects. Consequently, the most interesting question is whether we can modify OPTICS so that we can trade-off a limited amount of accuracy for a large gain in efficiency. Incrementally managing a cluster-ordering when updates on the database occur is another interesting challenge.

III. PROPOSED METHODOLOGY

3.1. EXISTING SYSTEM

DBSCAN is an effective density-based clustering method which was first proposed in 1996. Compared with other clustering methods, DBSCAN possesses several attractive properties. First, it can divide data into clusters with arbitrary shapes. For example, it can find clusters totally surrounded by another cluster. Second, DBSCAN does not require the number of the clusters a priori. Third, it is insensitive to the order of the points in the dataset. As a result, DBSCAN has achieved great success and become the most cited clustering method in the scientific literature. For

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2017

large-scale dataset analysis, MapReduce is a desirable parallel programming platform based on shared-nothing architectures.

Positive Aspects
It discover the clusters which are dense regions of data points and are separated by sparse regions with respect to given density parameters. Density-based clustering can discover clusters with arbitrary shapes.It uses ε-neighborhood for creating clusters of data points having density parameters. Using spherical ε-neighborhoods by DBSCAN is problematic when there are clusters with varying densities.

Negative Aspects
However, performing DBSCAN efficiently in real-world applications is challenging due to two reasons. First, the sizes of the datasets are growing rapidly so that they cannot be held on a single machine anymore; Second, the advantages of DBSCAN come at a cost, i.e., at a much higher computation complexity compared with other clustering methods such as K-means. A recommended way to solve these problems is to perform DBSCAN algorithm in parallel on a shared-nothing cluster**.**

Designing an efficient DBSCAN algorithm in MapReduce has three main challenges. First, due to the simplicity of MapReduce, the data interchanging mechanism is limited. Specifically, data transferring between map nodes or reduce nodes is not encouraged, making the parallelism of DBSCAN nontrivial. Second, although MapReduce can process text-based data queries efficiently, it becomes quite clumsy when dealing with spatio-temporal data. The main reason is that spatio-temporal data are multi-dimensional, yet MapReduce does not provide any mechanisms, such as R-tree or KD-tree, to improve the efficiency of multi-dimensional search. Third, maximum parallelism can only be achieved when the data is well balanced. However, in real applications, data are often highly skewed and thus cares should be taken during the process of data partitioning. In this paper, we address these challenges by proposing MR-DBSCAN, an efficient parallel DBSCAN algorithm using MapReduce.

3.2 PROPOSED SYSTEM
This work propose a serial density-based clustering algorithm for large data, called DBCURE, which discovers clusters with widely varying densities as well and is suitable for the MapReduce framework. DBCURE is more suitable to parallel with MapReduce than OPTICS.Similar to DBSCAN DBCURE also repeatedly performs the two-step approach. The first step chooses an arbitrary core point, which is an unvisited point in D, as a seed and inserts it to the seed set S. Then, in the second step, we retrieve all points that are density-reachable from the seed set S by extracting a point p from S, inserting its τ-neighborhood into S and repeatedly adding the τ-neighborhood of every core point p AS into S until S becomes empty. To discover the density-reachable points from the seed set S, we need to find the ellipsoidal τ-neighborhood points for a given point efficiently. We can prove that the discovered clustering result obtained by DBCURE satisfies the density-based cluster property. Since extending DBSCAN to DBCURE is straight forward.

Advantages
- ❖ It discovers the clusters of varying densities effectively.
- ❖ It scales up well with MapReduce framework.
- ❖ It is faster than DBSCAN and OPTICS.
- ❖ It calculates τ-neighborhood instead of Ɛ-neighborhood.

3.3. Problem Statement
DBCURE-MR: the parallel DBCURE algorithm using MapReduce While DBCURE finds each cluster one by one repeatedly, we design the parallel DBCURE algorithm, called DBCURE-MR, to discover all clusters together by expanding every core point in parallel.

The outline of our DBCURE-MR consists of the following four steps:
Step 1 – **Estimation of the neighborhood covariance matrices**:
Step 2 – **Computation of ellipsoidal τ-neighborhoods**:
Step 3 – **Discovery of core clusters**:
For a core point p, the set of p and its directly density-reachable points is called a core cluster. We obtain the list of the ellipsoidal τ-neighborhood points for every point in D by using each distinct point as the grouping key. For each grouping key, if the number of the ellipsoidal τ-neighborhood points is atleast δ, the point is a core point and we output the point with its ellipsoidal τ-neighborhood points.
Step 4 – **Merge of core clusters**:

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2017

Repeatedly merge the core clusters sharing a common core point to generate the final clusters. We next show an illustration of how DBCURE-MR works using the following example.

Steps	Algorithm	Output
Step1	COVMATX1- ↓ COVMATX2-	Statistics to estimate Covariance matrix. Neighborhood covariance matrix for every point.
Step2	MAXMBR-MR ↓ τ-NEIGHBOR-MR	Maximum width in the MBR of (∑,τ)-ellipses of all points All pairs of points each of which includes the other one in its τ-neighborhood
Step3	FINDKERNAL-MR	Every core cluster
Step4	MERGE-CLUST-MR	Final clustering Result

COVMATX-MR: computing neighborhood covariance matrices

To compute neighborhood covariance matrices quickly, we divide the space containing the points in D into a rectangular grid by partitioning each dimension into M intervals with the same length. We compute the neighbor- hood covariance matrices in all grid cells independently in parallel using COVMATX-MR which consists of COVMATX1-MR and COVMATX2-MR. Fig. 2. Neighboring cells and an MBR.(a)Weight of neighborhood points,(b) maximum bounding rectangle. Recall that Si½k1;…; kd is the sum of the i-th coordinates of every point in the cell Gk1;…;kd and Sij½k1;…; kd is the sum of products between the i-th coordinate and the j-th coordinate of every point in the cell Gk1;…;kd . Let g denote the cell ID which is represented by (k1;…; kd). COVMAT1- MR calculates Si½g_ and Sij½g_. For each point pi in the cell represented by the cell ID g, a map function is called and emits a key-value pair ⟨g; pi⟩. Then, for each g, a reduce function is called and outputs ⟨g; (Si½g_; Sij½g_)⟩. The map function in COVMATX2-MR gets as input both of the points in D and the output of COVMATX1-MR. If the input is a point pi in D, the map function computes the grid cell ID g of pi and emits a key-value pair ⟨g; pi⟩. If the input is the key-value pair ⟨g; (Si½g_; Sij½g_)⟩ output by COVMAT1- MR, the map function emits ⟨g′; (Si½g_; Sij½g_)⟩ for every neighboring cell ID g′ of g. Then, the reduce function called for key g computes the neighboring covariance matrix Σi of each point pi in the cell of g according to Eq. (5), and outputs ⟨pi; Σi⟩. Time complexities: To compute Si½g_ for every i in ½1; d_ and Sij½g_ for every pair of (i; j) in ½1; d_ _ ½1; d_, the reduce function of COVMATX1-MR takes O((n)n _ d2)=Md) time with n=Md data points in each grid cell. When s machines are available for executing reduce functions, the time complexity of COVMAT1- MR becomes O((n)n _ d2)=Md_ ⌈Md=s⌉). In COVMATX2-MR, we compute a neighborhood covariance matrix for each grid cell g by using Si½g′_ and Suv½g′_ from every neighboring cell g′. Since there are O(3d) neighboring cells for each grid cell, its time complexity is O(3d _ ⌈Md=s⌉).

τ-NEIGHBOR-MR: computing ellipsoidal τ-neighborhoods:

Next present the MapReduce algorithm T-NEIGHBOR-MR which discovers core points with their τ-neighborhoods. To perform similarity joins for finding τ-neighborhood points efficiently, we exploit the data structure called ε-tree proposed in [27,28] for efficient similarity joins in a traditional setting. To find every pair of points whose distance is at most ε, the space containing the points is divided into a uniform rectangular grid with width ε and we evaluate the distances between every pair of points where each one from its own grid cell and neighboring grid cells. However, to find an ellipsoidal τ-neighborhood, we cannot divide the space with the grid with a uniform width in every dimension. Thus, we compute the MBRs of the (Σi; τ) ellipses of all points and find the maximum width in each dimension of all those MBRs to use it as the grid width of the dimension in the ε-tree. Then, we can guarantee that every ellipsoidal τ-neighborhood of a point is always in its own grid cell or the neighboring grid cells. Computation of the widths of grid cells: For each point p_i and its covariance matrix Σ_i produced by COVMATX-MR, a map function is invoked to compute the MBR of (Σ_i; τ)_ellipse for pi introduced in Section 3.4. Let Li(ℓ) be the length in the ℓ-th dimension of the

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2017

MBR of the point pi. The map function emits the key-value pair ⟨ℓ; Li (ℓ)⟩ for every dimension ℓ. Then, a reduce function is called with each key ℓ and outputs the maximum length among the values in the input value list. Combine functions can be used to speedup this MapReduce phase before reduce functions are called. The output of this phase is the list of every dimension's maximum length in the MBRs of (Σ_i; τ)-ellipses for all points in D and the list is broadcast to map functions in the next MapReduce phase of similarity joins. We call the MapReduce algorithm MAXMBR-MR. Similarity join: We conceptually divided D into m partitions P1,…,Pm and we split the pairs of points in D by map function into the m _ (m)1)=2 partitions each of which is identified by(Pi, Pj) and contains all points in P_i and P_j. Then, for every partition of (P_i, P_j), a reduce function is called and finds all pairs of points each of which is within both of their ellipsoidal τ-neighborhoods using ε-trees. Finally, we perform the union with the similarity join results from all partitions.

Map function

Each map function takes a point pi and its covariance matrix Σi in the output of COVMATX-MR as input, and assigns a conceptual partition ID p_{id} to the pair (pi; Σ_i) by the hash function which generates an integer uniformly in the range of [1, m]. Then, for each point, the map function outputs the key-value pairs ⟨(i; p_{id}); (pi; Σ_i)⟩ for every i such that 1<=i<=p_{id}, and outputs ⟨(p_{id}; i); (p_i; Σ_i)⟩ for every i such that pid <= I <=m. Note that we always make sure that every key (pid_i, pid_j) generated from the map function satisfies the condition of pidi <= pid_j.

Reduce function

For each distinct key of the conceptual ID pairs (pid_i, pid_j), the list of the points emitted from map functions are generated by the shuffling phase in the MapReduce framework before reduce functions are invoked. When a reduce function is called for each distinct key (pidi,pidj), we have two cases: (1) pid_i = pid_j and (2) pidi ≠ pidj. When pidi=pidj, every point in the partition of pid_i appears exactly once in the lists once the map function emits every point in the partition pid_i only once with key (pid_i, pid_j). Thus, we can simply test whether, for every distinct point pair (pi, pj) in the partition pid_i, pi includes p_j in the (Σ_i; τ) ellipse of p_i and p_j also includes pi in the (Σj; τ) ellipse of p_j. If they include each other in their (Σ_i; τ) ellipses, we output both (pi, pj) and (pj, pi). To do this work quickly, we build an ε-tree with the points of pid_i and execute the self-join algorithm in [28]. When pid ≠ pidj, we find the pairs of points each of which includes the other within its ellipsoidal τ-neighborhood among all pairs of the Cartesian product between the points from pid_i and pid_j since the pairs of points from the partition pid_i (or pid_j) only are computed in the reduce function with key (pid_i; pid_i) (or (pid_j; pid_j)).

FINDKERNEL-MR: Discovering core clusters

Next we develop the MapReduce algorithm called FINDKERNEL-MR that discovers all core clusters. For each point pi, if the size of its τ-neighborhood is atleast δ, pi is a core point by Definition 3.1 and we output pi with its τ-neighborhood. Otherwise, we output nothing. For a core point pi, we refer to the set consisting of pi itself and the points in its τ-neighborhood as a core cluster. In FINDKERNEL-MR, a map function is called for each pair of points ⟨pi; pj⟩ produced by τ-NEIGHBOR-MR and emits the input pair ⟨pi; p_j⟩. The key-value pairs output by each map function are grouped by each distinct key point in the shuffling phase and the list of every point pi with its τ-neighborhood Nτ (pi) is generated. For each ⟨p_i; $N_\tau(p_i)$⟩, a reduce function is next invoked. If $|N\tau(pi)| + 1 \geq \delta$ (i.e., pi is a core point), it outputs ⟨pi; Nτ(pi)⟩. If |Nτ(pi)|+1<δ (i.e., pi is a border point),the reduce function outputs nothing. The result of FINDKERNEL-MR is the list of every core point with its τ-neighborhood. We call the resulting list the core cluster table. Time complexity: Since each reduce function of FINDKERNEL-MR computes a core cluster with each data point, it takes O (B _ ⌈n=s⌉) time where B and s denote the maximum size of a core cluster and the number of machines respectively.

MERGE-CLUST-MR: generating final clusters by merging core clusters

It generate the final clusters by merging the clusters sharing atleast a common core point in the core cluster table R obtained previously. We will first present the serial algorithm MERGE-CLUST and next develop the parallel algorithm MERGE-CLUST-MR.

MERGE-CLUST-ALL

After MERGE-CLUST-PARTIAL produces Sis and P, MERGE-CLUST-MR in Fig. 8 iteratively selects each Si as a pivot and invokes MERGE-CLUST-ALL to merge the clusters in every Sj (jai) to the clusters in Si. The pseudo code of MERGE-CLUST-ALL is presented below. At the i-th iteration of the for loop in MERGE-CLUST-MR, Si is broadcast to every reduce function before the execution of MERGE-CLUST-ALL. For each key-value pair ⟨u; v⟩ output by MERGE-CLUST-PARTIAL, a map function is called. If u € Di and v € Dj, the key-value pair ⟨j; (u; v)⟩ is emitted so that the reduce function invoked with key j can merge the cluster of v to that of u by updating Si and Sj. Then, for each distinct key j, a reduce function is invoked, but it also receives two disjoint set data structures Si and Sj where Si is

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2017

broadcast by the main function of MERGE-CLUST-MR and Sj is read from the distributed file system. If v is a core point, we merge the cluster of v to that of u in Si by using the function UNION-TO (v, c_u, S_j) which sets the representative node of v's cluster in S_j to point to c_u which denotes the node representing u's cluster.

If v is not visited yet, since v is a border point which should be merged into the cluster of u, not only the cluster of v is merged to that of u but also the status of v is set to BORDER. If MERGE-CLUST-ALL in each iteration is done, we write the data structure S_i on the distributed file system. While the reduce function with key j merges the cluster of u € Di to that of v € D_j, if any ancestor node is encountered while traversing the path from u to the root (i.e., representative) node, since we cannot update such ancestor nodes in the current execution of MERGE-CLUST-ALL, we output ⟨v; c_u⟩ where c_u is the first encountered ancestor node belongs to S_k ($k \neq j$) from u. We refer to such output of all ⟨v; cu⟩ s in the i-th iteration of the for loop in MERGE-CLUST-MR as F_i and we let $F = F_1 \cup \cdots \cup F_K$. All S_is and F will be processed together as the post processing step by MERGE-CLUST-FINAL.

Merge-Clust-Final
Merge the clusters, which could not be combined by MERGE-CLUST-ALL, by using MERGE-CLUST-FINAL. We call the serial function MERGE-CLUST- FINAL and its pseudo code is presented in Fig. 12. Note that all Sis and F produced by MERGE-CLUST-ALL are stored in the distributed file system. Each pair (u, v) in F implies that the cluster of v needs to be merged into the cluster represented by the node u. We first perform set-union operations based on (u, v) in F to simplify the updates to all Sis. For example, assume that (u, v) and (v, w) exist in F. If there is a node pointing to w in a Si, we have to update the node to point to u rather than v. Thus, before updating any Si, we perform set-union operations based on (u, v) in F and generate a disjoint-set data structure S_f in whose nodes consist of the distinct points appearing in F in lines 1–5. Then, while reading Si from disk one by one, for every node w in Si, we consult S_f in on whether to update the node or not. Suppose a node w points to the node v. If v exists in S_f in and the root node of v is u in S_f in, MERGE-CLUST-FINAL updates the node w to point to u.

IV. RESULT AND DISCRIMINATION OF THE BEST

This work conduct all the experiments on a 20-node cluster, each node has an Intel (R) Pentium CPU A1018 @ 2.10GHz and 2 GB RAM. The operating system used in node is Windows 7. All nodes were hosted in a single track. All algorithms were implemented using JavaC Compiler of version 1.5. We used ApacheHadoop 2.6.0 framework for MapReduce implementation. For the comparison of our experiments we use several algorithms to evaluate the performance of those algorithms with different inputs.

DBSCAN
DBSCAN finds clusters of arbitrary shape and able to find clusters in the dimensional spatial databases. It requires users to specify input parameters which can be a tedious task and may affect the clustering. And DBSCAN could not find the correct clusters in any setting of the density parameters with all data sets. When we increase the density parameters (Ɛ, δ) too many clusters will generated on sparse regions. Moreover DBSCAN was very sensitive to change of Ɛ.

OPTICS
OPTICS is an extension of DBSCAN and it works on the same way as DBSCAN. DBSCAN has the weakness of sensitivity on density-parameters. OPTICS overcomes this weakness by creating an ordering of points which automatically extracts clusters in data. The time complexity of OPTICS is similar to DBSCAN in case of indexing structure. But it cannot be parallelized with MapReduce.

DBCURE
It can discover clusters well with widely varying densities, similar to OPTICS which is difficult in nature to be parallelized using MapReduce, while the traditional DBSCAN algorithm cannot find them.

DBCURE-MR
It scales up with MapReduce framework and outperforms DBCURE-GRID-MR which utilizes the idea in the state-of-the-art parallel DBSCAN.

DBSCAN-MR
DBSCAN-MR which is obtained by parallelizing the traditional DBSCAN using our parallelization technique used in DBCURE-MR, is also much faster than DBSCAN-GRID-MR which is currently the state-of-the-art for parallelized DBSCAN.

UGC Approved Journal
IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2017

DBSCAN-GRID-MR
It represents the implementation of the state-of-the-art parallel DBSCAN using MapReduce.

DBCURE-GRID-MR
We implement another variant of parallel DBCURE utilizing the parallelization technique of using a grid. This simply performs COVMATX-MR before executing DBSCAN-GRID-MR and expands clusters by using the ellipsoidal τ-neighborhoods. From all the above experiments we can see that DBCURE is faster than DBCURE-MR when we use a single machine due to the overhead of the MapReduce framework. However, as the number of machines increases, DBCURE-MR becomes faster linearly. With 20 machines, DBCURE-MR is 5 times faster than DBCURE using a single machine. As a result our DBCURE-MR shows good speed up with the real-life data set.

V.CONCLUSION

In this paper, we study the problem in density based clustering in parallelization with MapReduce framework. As we study about traditional DBSCAN which is difficult in parallelization of clustering varying density data sets. Next we tryout this problem by OPTICS algorithm which overcomes the weakness of DBSCAN. However OPTICS is hard to parallelize. Thus, we developed a new variant from DBSCAN called DBCURE, which have the advantage to work in parallel with MapReduce, Hadoop and can be work well with varying density datasets. We next develop the parallel version of DBCURE called DBCURE-MR which gives the correctness proof for efficiency and accuracy. By the experimental results, we showed that our DBCURE-MR finds the good clusters efficiently even it is large in volume and having different densities and scales up well with MapReduce framework.

REFERENCES

[1] Amandeep Kaur Mann and Navneet Kaur, "Survey Paper on Clustering Techniques",IJSETR, 2278 – 779.
[2] Pavel Berkhin, "ASurvey of Clustering Data Mining Techniques", pp.25-71, 2002.
[3] Oded Maimon and Lior Rokach, "Data Mining AND Knowlwdge Discovery Handbook", Springer Science Business Media.Inc, pp.321-352, 2005.
[4] Han. J. and Kamber M., "Data Mining Concepts and Techniques", Morgan Kaufmann Publisher, 2001.
[5] K.Kameshwaran and K.Malarvizhi, "Survey on Clustering Techniques in Data Mining", IJCSIT, Vol. 5, 2014, 2272-2276.
[6] Aastha Joshi and Rajneet Kaur, "A Review: Comparative Study of Various Clustering Techniques in Data Mining",IJARCSSE, Vol. 3, 2013, 2277-128X.
[7] Manish Verma, Mauly Srivastava, Neha Chack, Atul Kumar Diswar and Nidhi Gupta, "A Comparative Study of Various Clustering Algorithms in Data Mining, "International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 3, pp.1379-1384, 2012.
[8] Pradeep Rai and Shubha Singh, "A Survey of Clustering Techniques", International Journal of Computer Applications, 2010.
[9] Anand V. Saurkar, Vaibhav Bhujade, Priti Bhagat Amit Khaparde , "A Review Paper on Various Data Mining Techniques", Vol. 4, Issue 4, pp.1379-1384, 2016.
[10] Dr.E.Kesavulu Reddy , "Comparative Analysis of Clustering Techniques in Data mining"
[11] Data Mining: Concepts and Techniques, Second Edition Jiawei Han University of Illinois at Urbana- Champaign Micheline Kamber
[12] DBSCAN-A Density-Based Spatial Clustering of Application with NoiseHenrik Bäcklund (henba892), Anders Hedblom (andh893), Niklas Neijman (nikne866)
[13] A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise Martin Ester, Hans-Peter Kriegel, Jiirg Sander, Xiaowei Xu Institute for ComputerS cience, University of Munich Oettingenstr. 67, D-80538 Miinchen, Germany
[14] DBSCAN CLUSTERING ON TOP OF MAP REDUCE FRAMEWORKSamata.S.Huddar, Pramod Department of Computer Engineering Visveswaraya technological University Bangalore, India, Department of Computer Engineering Visveswaraya technological University Bangalore, India
[15] The Theory of Dynamic Programming- Richard Bellman
[16] Fast Density Based Clustering Algorithm Priyanka Trikha and Singh Vijendra International Journal of Machine Learning and Computing, Vol. 3, No. 1, February 2013
[17] MapReduce: Simplified Data Processing on Large Clusters - Jeffrey Dean and Sanjay Ghemawat jeff@google.com,Google, Inc. Iterative big data clustering algorithms: a review Amin Mohebi1,*,†, Saeed Aghabozorgi1, Teh Ying Wah1, Tutut Herawan1 and Ramin Yahyapour2 1Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia 2Gesellschaft fur wissenschaftliche Datenverarbeitung mbH Gottingen (GWDG), Gottingen, Germany
[18] l-DBSCAN : A Fast Hybrid Density Based Clustering Method P. Viswanath, Rajwala Pinkesh Department of Computer Science and Engineering, Indian Institute of Technology - Guwahati, Guwahati - 781039, India.
[19] MapReduce: Simplified Data Processing on Large Clusters by Jeff Dean, Sanjay Ghemawat, Google, Inc.
[20] MRG-DBSCAN: An Improved DBSCAN Clustering Method Based on Map Reduce and GridLi Ma1, Lei Gu, Bo Li ,Shouyi Qiao, Jin Wang.
[21] MR-DBSCAN: An Efficient Parallel Density-based Clustering Algorithm using MapReduce Yaobin He, Haoyu Tan, Wuman Luo, Huajian Mao, Di Ma, Shengzhong Feng, Jianping Fan,Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China Graduate University of Chinese Academy of Sciences, Beijing, China Hong Kong University of Science and Technology, Clear Water Bay, Kowloon Hong Kong National University of Defense Technology, Changsha, Hunan 410073, China .
[22] MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithmfor heavily skewed data Yaobin HE , Haoyu TAN, Wuman LUO, Shengzhong FENG, Jianping FAN, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China .