An Efficient Parallel Clustering Algorithm for Large Scale Database

Jianfeng Yang, Puliu Yan, Yinbo Xie
School of Electronic Information, Wuhan University, Wuhan, Hubei, China
Email: {yjf, xyb}@whu.edu.cn
Qing Geng
Hubei Bureau of Surveying and Mapping, Wuhan, Hubei, China
Email: gq@whu.edu.cn
Jolly Wang, Nick Bao
PRC Education, Intel China Ltd. Shanghai, China
Email: {Jolly.wang,Nick.bao}@intel.com

Abstract— **In this paper, we propose a new parallel clustering algorithm, named Stem-Leaf-Point Plot Clustering Algorithm (SLPPCA). SLPPCA tends to produce clusters of different shapes and sizes, and according to our experiments, it can produces clusters more efficiently than traditional methods. SLPPCA can fully exploits the data-parallelism of data objects, and adopts a task decomposition design step to balance the workloads of multi-core processors to achieve a high speedup. We implemented SLPPCA to large scale data base on duo-core processor and quad-core processor based computer separately and analyzed its performance. The experimental results show that the clusters it produced were particularly good either in different density or shapes, furthermore, with the parallel pattern used in SLPPCA on multi-core platform, the speedup was almost linear with the numbers of cores in processor and the number of data points. Moreover, SLPPCA can generate satisfactory cluster number automatically in clustering process.**

Index Terms— **Clustering, SLPPCA, SLPP, Parallel Processing, Performance Analysis, Parallel Pattern**

I. INTRODUCTION

Clustering can be considered the most important unsupervised learning problem in data mining and has long played an important role in a variety fields such as molecular biology, geography, information retrieval, pattern recognition, KDD and etc,. Clustering analysis can groups data objects based only on information found in the data itself that describes the objects and their relationships. The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data, try to obtain the best result that objects in a group be similar to one another and different from the objects in others groups [1].

In general, Euclidean or cosine distance and density are used to measure the similarity among objects in clusters, but some characteristics of high-dimensional data base, for instance, the inherent sparsity of data objects, difference in shapes, size and density of clusters, cause the traditional clustering algorithms may have a difficulty in producing meaningful clusters, besides, sometimes the traditional clustering algorithms need user to provide some pre-cognitive parameters such as the cluster number k or the threshold ϵ of similarity of data objects or density in advance of the starting of algorithms, but it is impossible for user to provide those precognitive parameters well and truth if they lack of sufficient understanding of the data objects' distribution characteristic or the relationship among data objects' attributes [2].

At the same time, in data clustering area, how to design a parallel clustering algorithm is becoming a key problem now because that since 2006 [3] [4], Multi-core area is coming, Multi-core processors are now the norm, parallel computers are now the norm, therefor, in the design of clustering algorithms we are need to think parallel to dear with large scale data base and to achieve even better clustering performance [5].

In this paper, we consider parallel data clustering. Our interest in clustering stems from the need to mine and analyze heaps of unstructured Chinese text documents. Clustering has been used to discover "documents types" in sets of unstructured Chinese text documents and to summarize and label such collections. Clustering is inherently useful in organizing and searching large text collections, for example, in automatically building an ontology like Sohu!(www.sohu.com). Furthermore, Conceptual structure generated by clustering is akin to the "Table-of-Contents" in front of books [6]. Finally, clustering is useful for personalized information delivery by providing a setup for routing new information such as that arriving from newsfeeds and new scientific publications.

In this paper, as our main contribution, we propose a parallel clustering algorithm named SLPPCA on shared-memory multi-core processors. SLPPCA, the abbreviation of Stem-Leaf-Point-Plot, was derived from Stem-Leaf-Plot(SLP) [7]. All SLP that was built according to different dimension can integrate into SLPP by each points' value in each dimension. i.e., each SLP will contains a dimensional value of every point. And, in the shared-

This work was supported Supported by the National Natural Science Foundation of China under Grant No. 60673149, the National High-Tech Research, Development Plan of China under Grant Nos.863-2007AA01Z105 and in part by NSF of Hubei Province (2008CDB319) and Intel China Research Group grant project–Research On Service Discovery in Vehicular Ad hoc Network Based on Intel Atom Platform.

doi:10.4304/jsw.4.10.1119-1126

memory programming environments such as OpenMP, if a task-based decomposition has been done, the data decomposition is only driven by the needs of each task. For example, in the fist phase of SLPPCA, each process of building SLP can be seemed as a task. To the best of our knowledge, a parallel SLPPCA clustering algorithm and its implementation have not been reported in the literature.

We now brief outline the rest of paper. In section II we'll discuss the clustering types and cluster types, and some parallel clustering algorithms; Section III presents the SLPPCA carefully, from its data structure to parallelism design; By implementing this parallel algorithm, in Section IV we analytically show that the speedup and the scalability of our algorithm approach the optimal as the number of data points increase or the number of cores increase. In Section V we'll make a conclusion mark and include a brief discussion on future work.

II. Related Works

Cluster analysis is related to other techniques that are used to divide data objects into groups and is sometimes referred to as unsupervised classification [8]. There are several types of clusterings , here we usually call clustering an entire collection of clusters [9]. The most commonly discussed distinction among different types of clusterings is whether the set of clusters is nested or unnested, i.e., hierarchical or partitional [10]. A partitional clustering is simply a division of the set of data objects into non-overlapping subsets such that each data objects is in exactly one set - one cluster. Commonly, a hierarchical clustering permit clusters to have subclusters and these nested clusters are organized as a tree. Another classification of clustering types is whether a data objects can be assigned to a single cluster or multiple clusters: Exclusive, Overlapping or Fuzzy. Especially in fuzzy clustering, every object belongs to every cluster with a membership weight that is between 0 and 1, from absolutely does't belong to absolutely belongs, here clusters are treated as fuzzy sets.

Furthermore, there are also some different types of clusters. A mostly used type in data cluster is well-separated type. For this type of clusters, a cluster is a set of data objects in which each object is closer to every one in the cluster than to any object not in the cluster, in order to separate the clusters, sometimes a threshold ε is used. Well-separated clusters do not need to be globular, but can be have any shape. The second type of cluster is Prototype-based [11], the objects in a cluster are closer to the pre-defined prototype. We commonly refer to prototype-based clusters as center-based clusters, which tends to be globular. Another commonly used cluster type is density-based, such as DBSCN, here a cluster is a dense region of objects that is surrounded by a region of low density, this type of cluster is often employed when the clusters are irregular or intertwined, and when the noise objects and outliers are present. Also, there still exists an important cluster type – Shared-property [12], in this cluster type, we can define a cluster as a set of data objects that share some property.

Lots of clustering algorithms such as K-Means [13], DBSCAN [14], CURE [15] and so on have been proposed in the past years, however, most of these algorithms require user to pre-define some parameters such as the cluster number k or the distance threshold ε or the density threshold δ, unfortunately it is difficult or impossible for users to set up these parameters, and an improper preference sometimes will lead to wrong or bad result. Therefor, it is very important to set up these parameters just rely on data set itself, without any artificial interference.

In the past years there has been an increasing interest in parallel implementations of data mining algorithms. several authors also have proposed some parallel clustering algorithms. Reference [16] discuss parallel implementations of the single link clustering method on an SIMD array processor. Their parallel implementation of the SLINK algorithm does not decrease the $O(n^2)$ time required by the serial implementation [17], but a significant constant speedup factor is obtained. Reference [18] describe parallel partitioning clustering (the k-means clustering algorithm) and parallel hierarchical clustering (single link clustering algorithm) on an n-node hypercube and an n-node butterfly. Their algorithms run in $O(n \log n)$ time on the hypercube and $O(n \log^2 n)$ on the butterfly. Reference [19] has described several implementations of hierarchical clustering algorithms. His implementation of hierarchical clustering algorithm achieves (n)time on a n-node CRCW PRAM and $O(n \log n)$time on $\frac{n}{n \log n}$ node butterfly networks or trees. All these parallel clustering algorithms have the following drawbacks [20]:

1) They assume that all objects can reside in main memory at the same time.
2) They need a large number of processors (about the size of the data set) to achieve a reasonable performance.

In the literature, several parallel algorithms for mining association rules have been proposed recently [16] [21] [22] [23]. However, for many applications, especially for mining in large spatial databases, scalable parallel clustering algorithms are still in great demand.

In this paper, we present a parallel clustering algorithm SLPPCA which is based on SLPP for knowledge discovery in very large spatial databases. We use the shared-memory architecture, with multi-core processor based computers.

III. The algorithm SLPPCA

SLPPCA aims to find out each boundary points set $\{BP_{j,i}\}$ of each arbitrary shape cluster by SLPP and label each BPs in the SLPP as boundary point of a cluster, here j stands for the j^{th} point, and i represents the i^{th} cluster, then connect the points of each $\{BP_{j,i}\}$ set in turn and create some contour lines if the DB is 2-dimensional, furthermore, a contour plane or hyperplane will be produced for 3-d(dimensional) or even high dimensional DB. Finally, the points contained in i^{th} contour line will be

marked as Inner Points of i^{th} cluster and represented as $\{IP_{k,i}\}$, k being the total number of inner points of i^{th} cluster. So, every arbitrary shape clusters can be represented with its BPs and IPs as Formula (1) . i is the number of clusters, which is automatically computed out when each contour line is produced. For simply connected data objects, the number of contour lines is the number of clusters; otherwise the cluster number is depended on the number of outer contour lines for complex connected data objects. Completely, the i is worked out by SLPPCA itself and need none any transcendental parameters to be imputed all through the algorithm procedure.

$$\{C_i\} = \{BP_{j,i}\} \cup \{IP_{k,i}\} \quad (1)$$

In order to visually illustrate the SLPPCA, we use a two-dimensional data points set as example to explain what SLPP is and how it is constructed and works.

A. SLPP

The data objects contains $n = 24$ points as Table 1, and each point was labeled as P_n according to the input order.

TABLE I.
DATA BASE 1

P	1	2	3	4	5	6	7	8	9
x	1.0	1.2	1.2	1.3	1.4	1.7	1.9	2.0	2.2
y	2.5	2.6	2.5	2.3	2.7	3.7	3.5	3.6	3.5
P	10	11	12	13	14	15	16	17	18
x	2.7	2.8	2.8	3.1	4.5	4.7	4.7	4.7	4.8
y	3.3	2.9	3.1	3.2	3.3	3.3	3.1	2.6	3.5
P	19	20	21	22	23	24			
x	4.8	4.9	5.0	5.4	5.6	5.7			
y	2.7	2.5	2.6	2.7	2.8	2.6			

Stem-and-Leaf Plot (SLP) [7] is a display that organizes a set of data to show its shape and distribution. In deed the SLPP is extended by SLP with each point's attributes value added to the leaf. In a SLPP, each data value is split into a "stem" and a "leaf-point pair" which also contains points no. A SLPP for a specified data base include x-dimensional SLPP (x is the number of dimensions) and all SLPP inter-connected by the point no information. For instance, the stem of point $P_9(2.2, 3.5)$ of 1^{st}-dimensional is 2, the leaf-point pair will be displayed as $(2, 9)$, in the same way the stem of the 2^{nd}-dimensional SLPP is 3 and leaf-point pair is $(5, 9)$. Let's use DB1 as example to illustrate the procedure of constructing SLPP.

Firstly, we use EDA (Explored Data Analysis) [7] method to eliminate the outliers and determine the dimension the data scattered more as the main dimensional of SLPP. For instance, as DB1 shows, $max(x) - min(x) = 5.7 - 1 = 4.7 > max(y) - min(y) = 3.6 - 2.3 = 1.3$, so x dimension is feasible.

The second step is to determine the stem granularity, a stem contains too many data of its leaf will enlarge the difference of stems, so we must divide a stem to 2 or 5 intervals if the leaf value contains (1, 3, 5, 7), (0, 2, 4, 6, 8) or (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Here 1-stem contains (0, 1, 2, 3, 4, 5, 6, 7), we divide 1-stem into 2 parts and labeled with "$*, \bullet$", here "$*$" and "$\bullet$" represent the scope of $(0, 1, 2, 3, 4)$ and $(5, 6, 7, 8, 9)$.

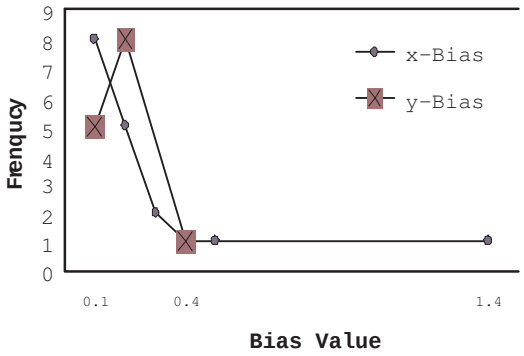

Figure 1. Bias Value

Finally, build the 2-d SLPP as Table 2 for the given DB, and computer out the bias value of each dimension. it means that the adjacent points at one stem should be attached to different clusters if the bias is bigger than the threshold ϵ. As Figure 1, the abscissa in a plane Cartesian coordinate system is the bias value and the y-axis is the times of the bias value appears, we choose the first minimized data as the threshold, here the x-bias is 0.4, the y-bias is 0.4, here we choose the max bias as the threshold and in this example, $\epsilon = 0.4$. Note that here the threshold ϵ was set by data itself, without any artificial interference.

By this time the SLPP of DB1 is produced, the right column contains the points of the same stem and points were set in ascend order with y-value, for example the leaf-point pair of $(3.3, 10)$ stem means the y-value of point P_{10} is 3.3. From Table 2, we can see that the SLPP do reflect the data distributions of the DB well and in truth.

B. SLPPCA

The flowchart of SLPPCA is described as Figure 2. After constructing the SLPP, the initial boundary points which labeled as boldface in Table 2 can obtained directly from SLPP and are put into $\{BP_j\}$ with repeated elements were eliminated out. The initial boundary points include the points in the first and the last positions of each leaf, and the adjacent points whose bias is big than threshold ϵ just as P_{19} and P_{16} are labeled as IBP also. So, the IBP set of DB1 is $IBP = \{P_1, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}, P_{20}, P_{19}, P_{16}, P_{18}, P_{21}, P_{22}, P_{23}, P_{24}\}$.

Each cluster is surrounded with BP_s that are contained in IBP set, we must eliminate the points out if those points are not the really BP. Each IBP item has at least 2^m (m being the dimensions, m=2 in DB1) adjacent objects, we define the most adjacent points $P_j(j = 1 - 2^m)$ of P_i are the points around P_i in 2^m regions, e.g., for 2-d DB, the $2^2 = 4$ regions of $P_i(x_i, y_i)$ are showed as Formula 2. Suppose that $\{P_j\}(j = 1 - 4)$ are the adjacent points set of P_i, if at least one point of $\{P_j\}$ were unfit

TABLE II.
IP SET OF DB1

x Stem	x:Leaf-Point Pair	y:Leaf-Point Pair (y-value, point no)
1∗	$(0,\mathbf{1})(2,2)(2,3)(3,\mathbf{4})$	$(2.3,\mathbf{4})(2.5,1)(2.5,3)(2.6,2)(2.7,\mathbf{5})$
1•	$(7,\mathbf{6})(9,\mathbf{7})$	$(3.5,\mathbf{7})(3.7,\mathbf{6})$
2∗	$(0,\mathbf{8})(2,\mathbf{9})$	$(3.5,\mathbf{9})(3.6,\mathbf{8})$
2•	$(7,\mathbf{10})(8,11)(8,\mathbf{12})$	$(2.9,\mathbf{11})(3.1,12)(3.3,\mathbf{10})$
3∗	$(1,\mathbf{13})$	$(3.2,\mathbf{13})$
3•		
4∗		
4•	$(5,\mathbf{14})(7,15)(7,16)(7,17)(8,18)(8,19)(9,\mathbf{20})$	$(2.5,\mathbf{20})(2.6,17)(2.7,\mathbf{19})\|(3.1,\mathbf{16})(3.3,14)(3.3,15)(3.5,\mathbf{18})$
5∗	$(0,\mathbf{21})(4,\mathbf{22})$	$(2.6,\mathbf{21})(2.7,\mathbf{22})$
5•	$(6,\mathbf{23})(7,\mathbf{24})$	$(2.6,\mathbf{24})(2.8,\mathbf{23})$

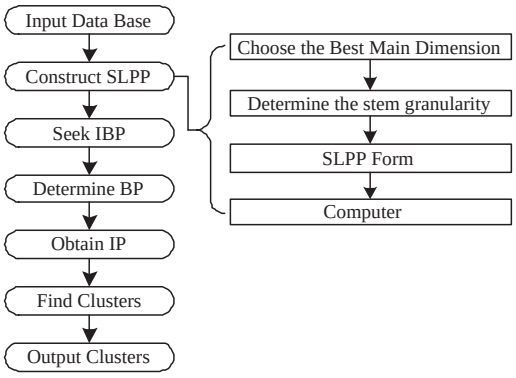

Figure 2. The Flowchart of SLPPCA

for Formula (3), then P_i is the BP of a cluster. Formula (3) can help to determine which point is a BP of IBP set.

$$R1:\begin{cases} x \geq x_i \\ y \leq y_i \end{cases} \quad R2:\begin{cases} x < x_i \\ y > y_i \end{cases}$$
$$R3:\begin{cases} x \leq x_i \\ y < y_i \end{cases} \quad R4:\begin{cases} x \geq x_i \\ y \leq y_i \end{cases} \tag{2}$$

$$\begin{cases} F_1(P_j) = x_j - (x_i + \epsilon) \leq 0 \\ F_2(P_j) = y_j - (y_i + \epsilon) \leq 0 \\ F_3(P_j) = -[x_j - (x_i - \epsilon)] \leq 0 \\ F_4(P_j) = -[y_j - (y_i - \epsilon)] \leq 0 \end{cases}, j = 1\tilde{}2^2 \tag{3}$$

When BPs are decided, the algorithm try to divide BP into k groups and labeled with $BP_i(i \in [1,k])$, i.e., the objects in DB1 will finally divided into k clusters, and the rest of the paper will prove that the k value is correct and the best.

Obviously,a cluster has more than 3 BPs, so we suppose that each BP_i set contains at least 3 points. From the 1^{st} object of $\{BP\}$ the SLPPCA try to search out the next most adjacent BP in one-way and this process will not stop until the BP is empty. When next adjacent BP back to P_{1st}, a BP_i set was produced and then took out from BP set, and a next search process will start form the 1^{st} object of BP. For DB1, the final BP set is described as Table 3. Then from SLPP, the IPs can be directly dug up as Table 4 shows. At each leaf or adjacent leaves,, the points between any pair of BPs which are belongs to the a same cluster are the Inner Points.

Finally, according to Formula (1), the cluster result is de-scribed in Scattered Plot as Figure 3. The result

TABLE III.
BP SET OF DB1

Set	Item
BP	$\{BP_1, BP_2, BP_3, BP_4, BP_5, BP_6\}$
BP_1	$\{P_1, P_4, P_5\}$
BP_2	$\{P_6, P_7, P_8, P_9\}$
BP_3	$\{P_{10}, P_{11}, P_{12}, P_{13}\}$
BP_4	$\{P_{14}, P_{16}, P_{18}\}$
BP_5	$\{P_{19}, P_{20}, P_{21}\}$
BP_6	$\{P_{22}, P_{23}, P_{24}\}$

TABLE IV.
IP SET OF DB1

Set	Item
IP_1	$\{P_2, P_3$
IP_2	ϕ
IP_3	ϕ
IP_4	$\{P_{15}, P_{17}\}$
IP_5	ϕ
IP_6	ϕ

contains 6 clusters in DB1. Suppose that k value equals to 2, 3 or 6 with K-Means clustering algorithm and the result is the best even that may be affected by the initial seeds selected at the beginning, the sum of squared error (SEE, Formula (4)) is described as Figure 4. When k equal to 6 the $SSE = 0.79$ is minimal, this result testify that the k number produced by SLPPCA is correct and the best.

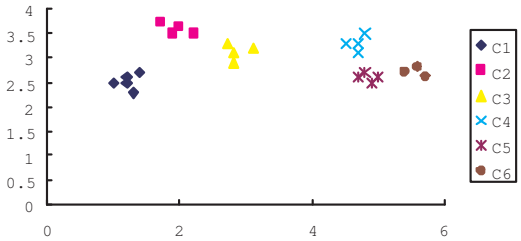

Figure 3. The clustering result of DB1(24P) by SLPPCA

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist(c_i, x)^2 \tag{4}$$

C. Clustering Performance

One obvious way to determine if a data set has clusters is try to cluster it. The clusters may be in arbitrary shape or different size or un-uniform density, furthermore, whether the result produced can reflect the natural number

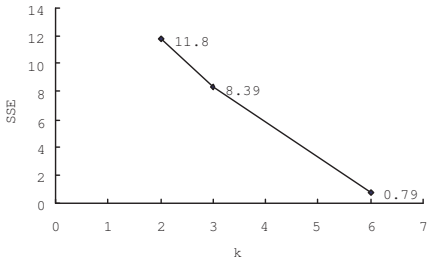

Figure 4. SSE versus k by K-means

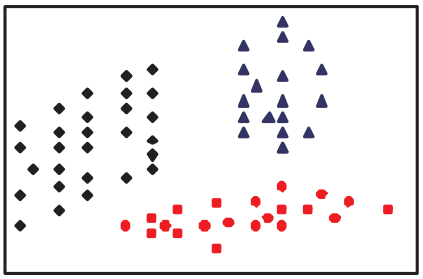

Figure 5. The Clustering Result of DB2 by SLPPCA

of clusters in a data set is still a big problem for most of the clustering algorithms, the SLPPCA whose idea and the process were presented in the previous sections can automatically find the best clusters number and fits for the clusters in different size and shapes. Since the main procedure of SLPPCA is to determine the boundary points sets only by SLPP and we need not to computer the Euclidean distance, the SLPPCA has a time complexity of $O(m)$ which was small than K-means algorithm, note that the m is the boundary points number while in K-Means, m is equal to the total objects number.

DB2 contains 70 points and the clustering result by SLPPCA is displayed as Figure 5, in the process the threshold $\epsilon = 0.5$, and the clusters number is 3 which were coincident with the natural number of clusters com from the graph-based view. From this example we can find that the SLPPCA can fit for non-globular cluster well just as C_2 in Figure 5. We compare K-means algorithm and SLPPCA use another data sets whose scatter plots are showed as Figure 6 and 7. DB3 as Figure 6(a) contains 205 points with clusters of different shape, DB4 as Figure 7(a) has 760 objects with clusters of different shape and density, and from the clustering result by K-means as Figure 6(c) and Figure 7(c) we can see that the K-means fails to identify clusters of non-spherical shapes (e.g., elongated), or clusters of different sizes for the reason that K-means can only be used for data that has a well-defined centroid such as a mean while the size and density can not affect the SLPPCA's result, the K-means uses a prototype-based notion of a cluster and SLPPCA uses graph-based concept, SLPPCA makes no assumption about the distribution of the given data and leave the data deliver the truth by SLPP, seeks BPs and IPs of each cluster from SLPP and finally produce k clusters automatically. The clustering result by SLPPCA for DB3

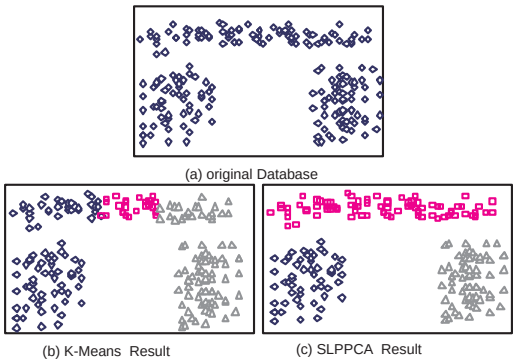

Figure 6. The cluster result of DB3 by SLPPCA(205P).

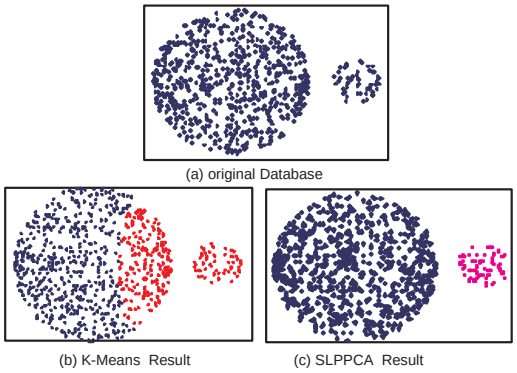

Figure 7. The cluster result of DB4 By SLPPCA(760P).

and DB4 presented in Figure 6(b) and 7(b) show that the SLPPCA perform well.

D. Automatically determination of k

For most of clustering algorithms, some parameters need to be specified in advance, e.g., the number of clusters k in K-means, distance threshold Eps and points numbers threshold $MinPts$ in DBSCAN [14], shrink factor α in CURE [15], and etc., while SLPPCA uses SLPP to discovery the distribution of data set and acquire some parameters without any interference by user all the time.

One of the novel aspects of SLPPCA is to determine boundary point sets by SLPP from graph-based view and find clusters rather than use the objects similarity (e.g., distance, density) as the evaluation principle to discovery clusters. i.e., the SLPPCA try to search out the contour lines of DB and produce clusters according to those contour lines found. The second contribution of the paper is to use SLPP to discovery the data attributes, SLPP is a extend form of Stem-and-leaf plot, which can reflect directly some characteristics include data congregation, dispersion (normal or skewness), outliers, spread and the cluster shape.

To study the effectiveness of SLPPCA for clustering data sets, we conducted extensive experiments and our result confirm that the quality of clusters produced by SLPPCA s much better than those found by some other existing algorithms, and with more lower time complexity.

IV. Parallelism Analysis of SLPPCA

The key to parallel clustering algorithm design is *exploitable concurrency*. Concurrency exists in a computational problem when the problem can be decomposed into subproblems that can safely execute at the same time [26]. In order to finding these potential concurrency, the parallel pattern language should be used to help in designing the parallelism of SLPPCA. As shown in Figure 8, the pattern language [24] is organized into four design space-*Finding Concurrency*, *Algorithm Structure*, *Support Structure*, and *Implementation Mechanisms*, which form a liner hierarchy, with *Finding Concurrency* at the top and *Implementation Mechanisms* at the bottom.

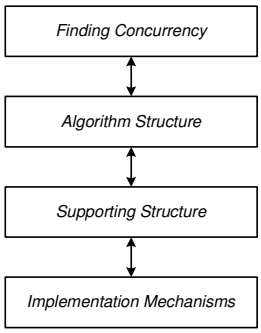

Figure 8. Overview of the pattern language.

Commonly, the *Finding Concurrency* design space is concerned with structuring the problem to expose exploitable concurrency. This is the most important level of parallelism design, at this level, we should focus on high-level algorithmic issues and reason about the problem to expose potential concurrency. Only concurrency has been found at this level, the rest levels can have works to do – how to tale advantage of these potential concurrency. i.e., the *Algorithm Structure* will focus on how to use these concurrency, and the *Support Structure* design space and the *Implementation Mechanisms* design space will focus on the programming pattern and how the patterns of the higher-spaces are mapped into particular programming environments.

A. Finding concurrency in SLPPCA

The *Finding Concurrency* design space will help us identify and analyze the exploitable concurrency in clustering problem. As shown in Figure 9, we will focus on

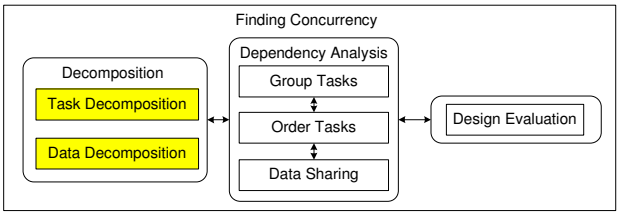

Figure 9. Overview of the Finding concurrency design space.

the decomposition. There are two decomposition patterns, *Task Decomposition* and *Data Decomposition*, are used to decompose the target problem into subproblems that can safely execute currently. We can think of this decomposition as occurring in two dimensions.

- The *Task-decomposition dimension* views the problem as a steam of instructions that can be broken into sequences called *tasks* that can execute simultaneously.
- The *Data-decomposition dimension* focus on the data required by the tasks and how it can be decomposed into distinct chunks.

Indeed, our SLPPCA are inherent parallelism. Since the SLPPCA is designed based on the SLPP, and all SLPP forms are interconnected by each points, therefor, we can build each SLPP (note that the number of SLPP forms is the dimensions of the data objects) separately. To find clusters, SLPPCA starts with the constructing of SLPP forms, as shown in Figure 10, we can see that the SLPPCA process can be designed well in parallel: the task of constructing SLPP can be paralleled, the tasks of seeking IBP, determining BP and obtaining IP can also be paralleled well. From the analysis before, we know that all these tasks are with high computation, are the ”hot spots” in the algorithm, so the parallel programming of these tasks will improve SLPPCA clustering performance efficiently.

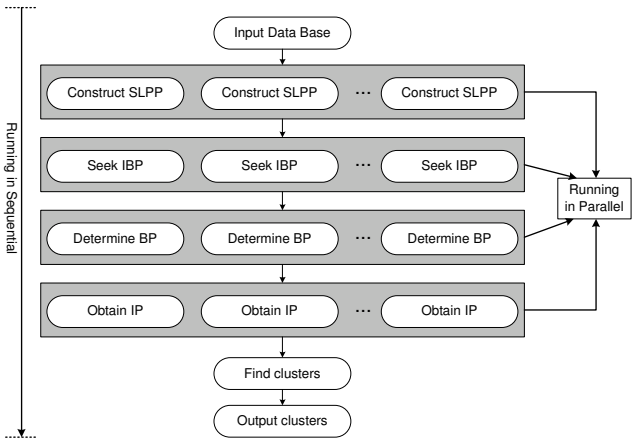

Figure 10. Parallelism in SLPPCA

Furthermore, we can see from Figure 11, the task of constructing SLPP can be paralleled in depth. This kind of nested parallel can help archive even high speedup.

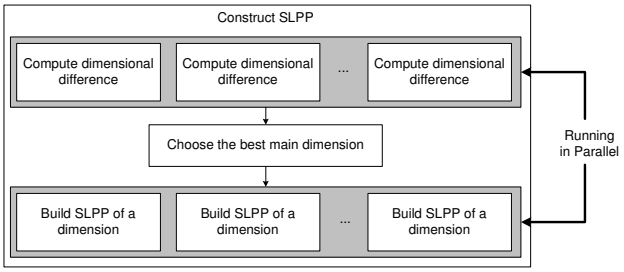

Figure 11. Parallelism in constructing SLPP.

The algorithm SLPPCA is sketched in Figure 12.

```
Algorithm SLPPCA (D)
//Precondition: all objects in D are unclassified.
//Note that there are none any more parameters needed to input.
Do Parallel
{
   Do Parallel
        {
            Construct SLPP forms for each dimension.
        }

   Seek IBP;
   Determining BP;
   Obtain IP;
}
Find Clusters and label every objects;
Output Clusters;
END;
```

Figure 12. Algorithm SLPPCA.

B. Experiment

In this section, we evaluate the performance of SLPPCA with respect to its scaleup, speedup and sizeup. We run our SLPPCA clustering algorithm on duo-core and quad-core processor based platform separately to evaluate the parallelism efficiency. The program was coded in C++ and Intel Threading Building Blocks (Intel TBB) [25] – which is the C++ template library for parallelism that extends C++ by abstracting away thread management and allowing straightforward parallel programming. And with TBB used in the experiments, we only need to focus on how to specify tasks and let the library map these tasks onto threads automatically.

Firstly, we'll compare the efficiency of SLPPCA between sequential version and parallel version program with objects as shown in Figure 7, on duo-core and quad-core based platform respectively.

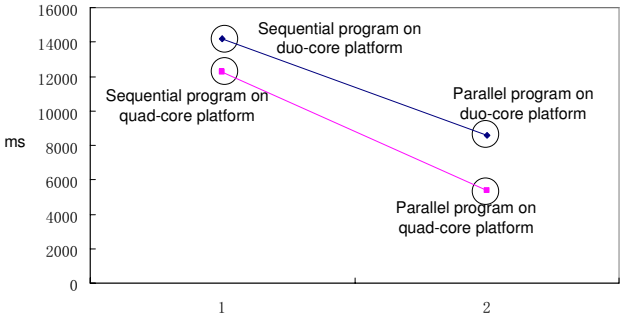

Figure 13. Comparison between sequential and parallel version program

Secondly, with the same data base, the parallel version program will be run on different platform to evaluate the speedup.

Thirdly, according to quad-core platform, we increase the data object number of the data base to check the parallel version program's scaleup.

From Figure 13 we can find that:

1) According to the same data base and same computing platform, the parallel version program will gain better performance than sequential version program. It is undoubtable that this gains come from the tasks decomposition and the concurrency of these tasks.
2) According to the same computing platform and parallel version program, the program running on quad-core processor based platform will achieve better performance than duo-core platform linearly. Obviously, in a sense, this is a good scaleup performance for numbers of cores.

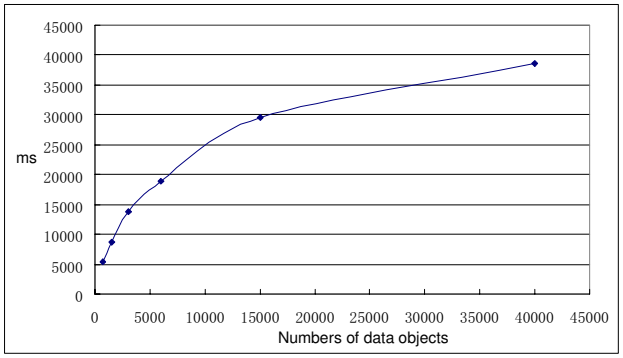

Figure 14. Scaleup of parallel version program

Also, according to our experiments result as shown in Figure 14, we can find that with the increase of data objects number in data base, the executing time increase linearly, the speedup is linearly. Therefor we can draw a conclusion that this kind of clustering algorithm - SLPPCA, with parallel version of program, can gain good scaleup performance. However, someone may argue that, along with the increase of data objects, the executing time should increase even larger than now, surely it is reasonable to think so, but in SLPPCA, because that even the number of objects increase, the bound points may unconverted or change so little, the algorithm just need to scan these increased objects to determine whether it is a bound point or not in the step of constructing SLPP, and in the rest of phases, these increased objects will not increase the computation much more.

V. Conclusion

In this paper, we proposed a parallel cluster algorithm SLPPCA for shard-memory multi-core processor based platform. By comparing with traditional cluster algorithms, SLPPCA not only can determine the cluster number k automatically, but also it can produce clusters with different density and shapes, more over, SLPPCA use EDA technology to set up some parameters totally rely on data itself, without any artificial interference. The clusters it produced can reflect the nature attribute of data in much better manner.

With the parallel pattern design language used in SLPPCA parallel version program design, we find that SLPPCA is parallel inherently. most of steps of SLPPCA can be paralleled well and the nested parallel help achieve even better performance. From the analysis and experimental results, a conclusion can be made that the SLPPCA can obtain good speedup and scaleup.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their useful comments and suggestions on how to improve this paper. Thanks so much to Prof. Chengcheng Guo, Delin Xia, Hao Jiang and Jing wu for their suggestions and comments.

REFERENCES

[1] Jiawei Han, Micheline Kamber, "Data Mining: Concepts and Techniques", *Morgan kaufamnn*, San Francisco, Auguest 2000.
[2] Kargupta, H., Hamzaoglu, I., Stafford, B., Hanagandi, V., Buescher, K., *ADMA:Parallel data mining agents for scalable text classification.*In: Proceedings of the High Performance Computing, Atlanta, GA, USA. (1997) 290C295
[3] Intel Processors, http://www.intel.com/products/processor.
[4] AMD Dual-Core Processors, http://multicore.amd.com/us-en/AMD-Multi-Core.
[5] T. Kohonen , *The self-organizing map*Proc. IEEE, vol. 78, no. 9, pp.1464C1480, Sep. 1990.
[6] W. Kent and A. Zahler, *Conservation, regulation, synteny, and introns in a large-scale C. BriggsaełC. elegans genomic alignment*Genome Res., vol. 10, pp. 1115-1125, 2000.
[7] Glenn J. Myatt, *making Sense of Data: A practical Guide to Exploratory Data Analysis and Data Mining*. Wiley-Interscience, november 28, 2006.
[8] M.S. Aldenderfer and R.K. Blashfield, *Cluster Analysis*, Sage Publicaitons, Los Angeles, 1985.
[9] Pang-Ning Tan, Michael Steinbach, et., *Introduction to Data Mining*,ISBN:0321321367, Addison-Wesley, 2006.
[10] S. C. Madeira and A. L. Oliveira, *Biclustering algorithms for biological data analysis: A survey*IEEE/ACM Trans. Computat. Biol. Bioinformatics, vol. 1, no. 1, pp. 24C45, Jan. 2004.
[11] R. Sun and C. Giles, *Sequence learning: Paradigms, algorithms, and applications*in LNAI 1828, . Berlin, Germany, 2000.
[12] Park, J.-S., Chen, M.-S., and Yu, P.S., *An effective hash based algorithm for mining association rules.*Proc. ACM SIGMOD Int. Conf. on Management of Data. San Jose, CA, pp.175C186.
[13] J B. MacQueen: "Some Methods for Classfication and Analysis of Multivariare Obervations", *Proceddings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1: 281-297, 1967.
[14] M.Ester, H-P. Kriegel, J.Sander, and X.Xu, A Density-Based Algorithm for Discoverying Clusters in Large Spatial Database With Noise, In *proc. of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 226-231, portland, Oregon, Auguest 1996, AAAI press.
[15] Sudipdo Guha, Rajeev Rastogi and Kyuseok Shim, CURE: An Efficient Clustering Algorithm for Large Database, In *prceedings of the ACM SIGMOD Conference on Management of Data*, 1998, Seattle WA, USA.
[16] Agrawal, R. and Shafer, J.C., *Parallel mining of association rules: design, implementation, and experience.*,1996. IBM Research Report.
[17] C. Pizzuti and D. Talia, *P-AutoClass: Scalable parallel clustering for mining large data sets*IEEE Trans. Knowl. Data Eng., vol. 15, no. 3, pp. 629C641, May-Jun. 2003.
[18] Li, X. and Fang, Z, *Parallel clustering algorithms*, Parallel Computing, 11:275C290.
[19] Li, X. and Fang, Z, *Parallel algorithms for hierarchical clustering*, arallel Computing, 21(8):1313C1325.
[20] R. Sharan and R. Shamir, *CLICK: A clustering algorithm with applications to gene expression analysis*in Proc. 8th Int. Conf. Intelligent Systems for Molecular Biology, 2000, pp. 307C316.
[21] Cheung, D.W., Han, J., Ng, V.T., Fu, A.W., and Fu, Y., *A fast distributed algorithm for mining association rules.*Proc. Int. Conf. on Parallel and Distributed Information System (PDIS96). Miami Beach, FL, USA.
[22] E.M. Rasmussen and P. Willett, *Efficiency of hierarchical agglomerative clustering using the ICL distributed array processor*, J. Documentation, 45(l) (Mar. 1989) l-29.
[23] Mehta, M. and DeWitt, D.J., *Data placement in shared-nothing parallel database systems*VLDB Journal,6:53C72
[24] Timothy G. Mattson, Beverly A. Sanders, Berna L. Massingill, *Pattern for parallel programming*,Addison-Wesley, ISBN:0321228111,Boston, 2005.
[25] James Reinders, *Intel Threading Building Blocks*,O'Reilly, Sebastopol, ISBN: 0596514808, 2007.
[26] H.Wang,W.Wang, J. Yang, and P. Yu, *Clustering by pattern similarity in large data sets*in Proc. ACM SIGMOD Int. Conf. Management of Data, 2002, pp. 394C405.

Jianfeng Yang was born in April 1976, Jiangsu Province, China. Now he is a teacher of School of Electronic Information, Wuhan University, P.R.China. His research interests include embedded system, multi-core programming, network, data mining and intelligent information processing.

Puliu Yan was born in Auguest 1963, Wuhan city, Hubei Province, China. Now she is a Professor and doctor supervisor at of School of Electronic Information, Wuhan University, P.R.China. Her research interests include network, data mining and intelligent information processing.

Yinbo Xie was born in October 1977, Hubei Province, China. Now he is a teacher of School of Electronic Information, Wuhan University, P.R.China. His research interests include embedded system, multi-core programming and network.

Qing Geng was born in December 1980, Hubei Province, China. Now she is an engineering at Hubei bureau of surveying and mapping, Hubei province, P.R.China. Her current research interests include network and data mining.

Jolly Wang was born in April 1975, Beijing, P.R.China. Now she is the manager of Intel PRC high education program. Her research interests include embedded system and Multi-core programming.

Nick Bao was born in September 1977, Shanghai, P.R.China. Now he is the manager of Intel software and service group. His research interests include Embedded system, Multi-core programming and Moblin.