Parallel Algorithm for the Chameleon Clustering Algorithm using Dynamic Modeling

Rajnish Dashora
SCSE
VIT University,Vellore

Harsh Bajaj
SCSE
VIT University,Vellore

Akshat Dube
SELECT
VIT University,Vellore

Geetha Mary .A
SCSE
VIT University,Vellore

ABSTRACT

With the increasing size of data-sets in application areas like bio-medical, hospitals, information systems, scientific data processing and predictions, finance analytics, communications, retail and marketing, it is becoming increasingly important to execute data mining tasks in parallel. At the same time, technological advancements have made shared memory-parallel computation machines commonly available to various organizations and individuals. This paper analyzes a hierarchical clustering algorithm named chameleon clustering which is based on dynamic modeling and we propose a parallel algorithm for the same. The algorithm utilizes the concept of parallel processors available and hence reduces the time to generate final clusters.

General Terms

Enhancement, Utilization, clustering, similarity, Algorithms et. al.

Keywords

Multicore Processors; Data Mining; Cluster analysis; Hierarchical Clustering; Chameleon; Data points; Shared Memory; Symmetric Multiprocessing(SMP);Dynamic Modeling; ParMetis .

1. INTRODUCTION

Cluster analysis can be used in market research, business, land use, biology, atmospheric research, astrology, web based applications plant observation and load analysis in power systems. Clustering is the process of grouping elements into classes or groups based on their similarity of constituent data. Hierarchical form of clustering is known as Hierarchical clustering and Agglomerative means to group the similar datasets into one cluster.

Existing algorithm only use a static model of clustering and not the information of individual clusters when they are merged. CURE (Clustering Using Representatives) and other algorithms ignore the aggregate interconnection of the item sets of two clusters[7].As CURE uses only representative points from each cluster and merges only the closest pair of representative points In this way it does not consider other points (data objects) in the clusters. However ROCK (Robust Clustering algorithms) has an advantage over CURE [15] as it takes the aggregate interconnectivity of the two clusters but it ignores the information about their inter-closeness [11] .

Chameleon clustering is an algorithm that uses dynamic modeling in hierarchical clustering. It does not depend on user supplied information but it automatically adapts to the internal characteristics of the clusters being merged [3].To know the internal characteristics it uses relative inter-connectivity and relative closeness of the two clusters[4] .Furthermore to make the algorithm to perform dynamically it uses the concept of K-nearest neighboring graph as in this algorithm the neighborhood radius of a data objects is defined by the density of region in which the object lies.

2. RELATED WORKS

PCURE [7] is the parallel implementation of CURE which makes an effective utilization of shared memory architecture on small-scale symmetric multiprocessors as well as high performance processors. PCURE uses static modeling in hierarchical clustering [6]. The algorithm uses per cluster information by using the index of the closest cluster in minimum distance to it. But the maintenance of this cluster information with larger index needs more computation time.

A parallel K-NN algorithms [2] that exist uses truncated bi-tonic sort. This sort has nlog2(k)/4 comparisons. The Euclidean metric takes a lot of portion of search time so this algorithm uses CUBLA on GPUs. CUBLA [14] is a library developed by NVIDIA for algebra calculations like dot product, matrix manipulations, vector products etc.

The chameleon algorithm for clustering works in 3 phases.

A) Getting sparse graph for the data using.

K-Nearest Neighbour Algorithm

B) Partition the Sparse graph using multi graph partitioning algorithms which are predefined in hMeTis Library[13] for graph partitioning algorithms to get smaller clusters depending upon their similarity.

C)In the third phase the algorithm merges the smaller clusters to remove the noisy data and make the clusters with significant size.

K-nearest neighbor is a supervised algorithm used to find similarity between data. It assigns a class-label to data values and then finds the ones that belong to the same class, and, connect them with an edge.

The third phase of merging of the clusters is done on the basis of their similarity to get final clusters. Similarity is measured with the two parameters.

i) Inter-connectivity

ii) Closeness

Inter-connectivity refers to the connection between two nodes and closeness is defined as the similarity between two nodes in two different clusters. Relative Inter-connectivity [3] is the connection between two nodes of two different clusters

For the two clusters the Relative inter-connectivity and Relative closeness is calculated using the formulas as follows for the two clusters Ci and Cj ,

Relative inter-connectivity,

$$RIC = \frac{Absolute\_IC(C_i, C_j)}{(internal\_IC(C_i) + internal\_IC(C_j))/2}$$

Where, $Absolute\_IC(C_i, C_j)$ = sum of weights of edges that connect C_i with C_j.

$internal\_IC(C_i)$ = weighted sum of edges that partition the cluster into roughly equal parts.

While, internal closeness of the cluster is the average weight of the edges in that particular cluster.

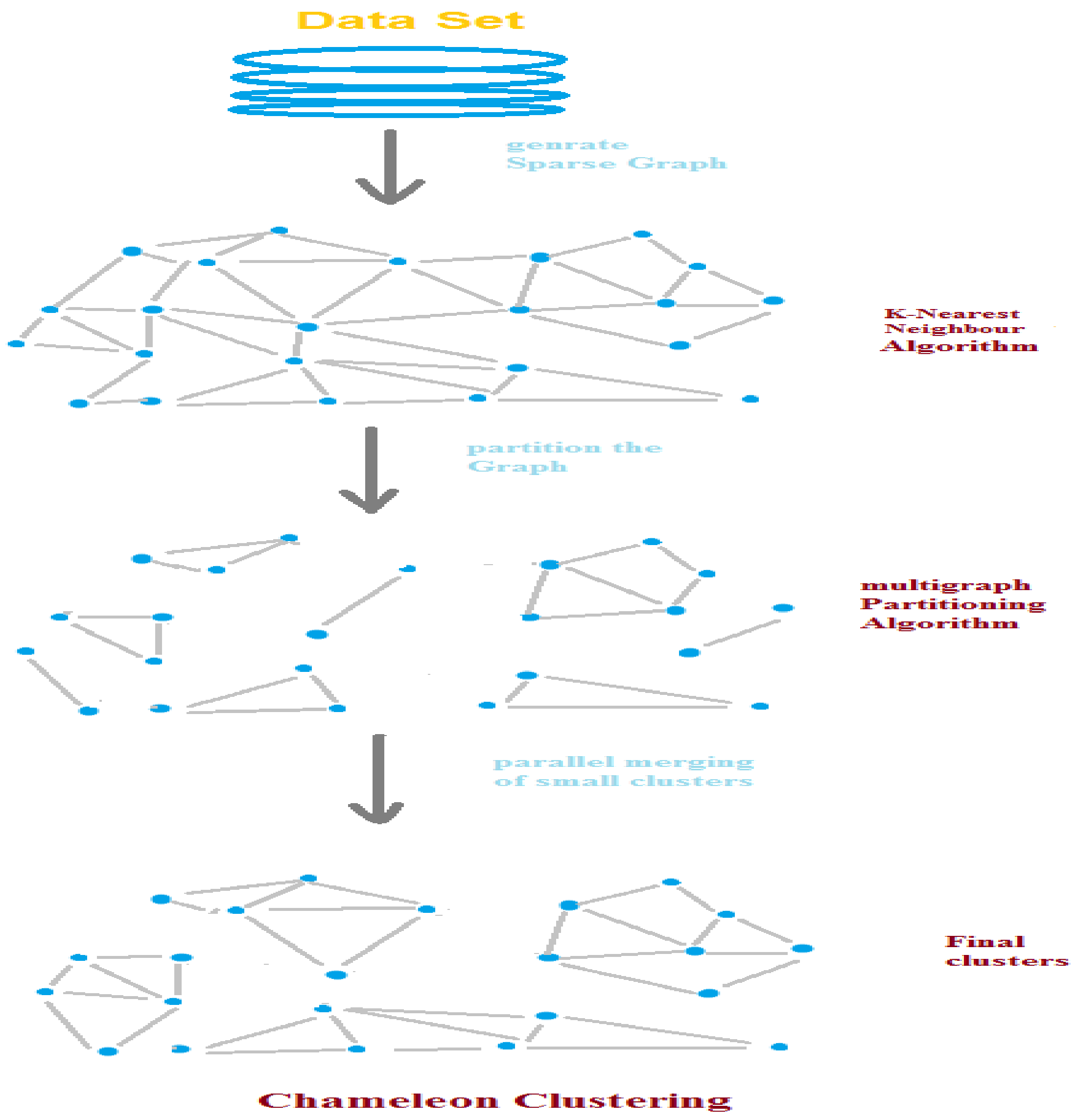
Data Set
genrate Sparse Graph
K-Nearest Neighbour Algorithm
partition the Graph
multigraph Partitioning Algorithm
parallel merging of small clusters
Final clusters
Chameleon Clustering

Fig 1: Chameleon Clustering

Relative closeness of two clusters is defined as

$$RC(C_i, C_j) = \frac{\bar{S}_{EC|C_i, C_j|}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{EC_{C_j}}}$$

Where,

SEC { C_i , C_j } is the average weight of edges that belong to a cut-set of the cluster C_i and C_j .

SEC C_i is the average weight of edges that belong to a minimum cut of the cluster C_i

| C_i | is the number of nodes in Cluster C_i .

Two clusters are merged if they have high value for the product of Relative inter-connectivity and Relative closeness. Sometimes, a user can assign a higher priority to a particular parameter using α.

$$RI(C_i, C_j)^{\alpha} * RC(C_i, C_j)$$

For the two clusters if the above function holds value greater than or equal to the threshold value for similarity then the clusters are merged.

If α > 1 then more priority is given to Relative inter-connectivity else if α < 1 Relative closeness gets more priority. Essentially, in case of α=1 both of them have same priority.

Pseudo code for merging algorithms for final clusters

```
1    RI – Relative inter connectivity
2    RC – Relative Closeness
3    α - user defined parameter
4    β – RI x RC
5    th – threshold value to take merging decision
6    n be number of clusters to be merge Algorithm:
7    for i=0 ... n  // i and j are used for clusters
8    for j=i+1 ... n
9    merge(i,j);
10   End for  // iteration j
11   End for  // iteration i
```

Merge function used above can be implemented as

```
1    merge($C_i$ ,$C_j$)
2    Calculate RI for $C_i$ and $C_j$
3    Calculate RC for $C_i$ and $C_j$
4    Calculate β = RI^α x RC
5    if β greater than or equal to th
6    merge the two clusters $C_i$ and $C_j$.
7     else
8    Do not merge the two clusters
9    end if
```

Disadvantages of chameleon includes,

a) Complexity**:** Algorithm has highly complex computation because it performs inter-connectivity and inter-closeness between all the data objects and this leads to a high computational complexity on a single processor.

b) It cannot recover from database corruptions.

3. PROPOSED SOLUTION

In this paper algorithm is proposed to reduce time complexity of chameleon algorithm. The exhaustive search time on K-Neighboring algorithm is also optimized. Parallel merging algorithm to merge datasets based on similarity has also distributed computation across threads. Therefore, it makes it scalable for larger datasets and gives a highly robust computation as one part of dataset doesn't affect the other. It also improves the complexity of sorting as heap sort is used. Sorting helps to keep similar item sets together [7].

Phase 1: Parallel K-Nearest Neighbor

This algorithm reduces time complexity of analysis. It reduces the search time by finding similar data and grouping them together.

Given a repository it analyses data and then compares the training data tuple with test tuple to get similarity of data value. This metric is known as Matric. Their metric is stored as a value of similarity between the two tested for tuple values. For the parallel implementation following procedure is followed:-

1) Sort the dataset based on Euclidean metric as,

a) Keep root node as the training tuple

b) Based on the value of Euclidean metric associated with a tuple put it in max-heap.

2) Generate max-heap with similar data as siblings.

3) A single thread out of the thread pool searches the data which will be in a single group and divide these data among other threads in pool.

4) If in step 3 the number of data chunks formed out of similarity sort is more compared to number of threads then keep them in queue.

5) Then whenever the threads are free de-queue the elements.

6) Threads will calculate Euclidean distance between all other data tuple and the data tuples allocated to it.

7) Find the Euclidean distance between all data tuples and connect them to form graph.

Connect the graph by taking value of Euclidean distance from the local variable of the thread to which that tuple was allocated.

8) Repeat step 7 for all the different tuples.

Phase 2: Partition the sparse Graph

After getting the sparse graph from the parallel k-NN method the graph is to be partitioned using multilevel graph partitioning algorithms [8] (using ParMETIS - Parallel Graph Partitioning and Fill-reducing Matrix Ordering [13]).

In the sparse graph an edge-cut between two points represents the similarity among them more the weight of the edge cut implies more similar the two points are.

Partition the graph such that the edge cut is minimized by finding the minimum edge cuts and removing them from the graph.

The above action will divide the sparse graph in smaller size clusters on the basis of their similarity.

Phase 3: Merging the clusters to get final clustering

As the clusters obtained by the partitioning of the sparse graph are very small and might have noisy data[9],clusters are merged to form larger sized clusters on the basis of two factors Relative inter-connectivity and Relative closeness of the clusters.

Parallel algorithm for merging the clusters:
The parallel threads are being used to perform the merging of the two clusters. Parallelism is achieved using work pool analogy.

Here merging of two clusters on basis of their similarity is a task or work assigned into work pool.

Work pool is the set of these independent tasks, functions or methods which can be executed in parallel.

Threads which are not executing any process will be assigned with a task from the pool of work. The task is of merging two clusters, defined by the function merge(C_i , C_j).

PSEUDO CODE

Algorithm:

```
1   void heapsort (array_of_nos, int n)
2   {
3   buildHp(array_of_nos,n);
4   shrinkHp(array_of_nos,n);
5   }
6   void buildHp (array_of_nos,n)
7   {
8   loop the three steps bellow till all nodes are
    checked;
        b.  End for // iteration i
```

Merge function used above can be implemented as

```
1   merge($C_i$, $C_j$)
2    Calculate RI for $C_i$  and $C_j$
3    Calculate RC for $C_i$  and $C_j$
4   Calculate $\beta = R I^{\alpha} x RC$
5   if $\beta$ greater than or equal to th
        a.  merge the two clusters $C_i$ and $C_j$. .
6   else
        a.  Do not merge the two clusters
7   end if
```

Pseudo code of parallel merging algorithms for final clusters

```
9   chld = i-1;
10  prnt = (chld-1) / 2;
11  make maximum of children as parents
12  }
13  void shinkhp(array_of_nos,n)
14  {
15  //here  each thread is assigned to a particular
    parent node
16  prnt=0;    //start from root
17  compare left and right child and make maximum as
    parent;
18  take the max heap from each thread thereby getting
    each parent node ;
19  i.e the nodes having right and left child ;
20  knowing the position of these set of nodes construct
    others;
21  }
22  levelorder()
23  {
24  traverse heap in level-order by dividing these levels
    to threads;
25  connect only siblings to form a graph ;
26  }
```

Parallel K-NN clustering with heap sorting algorithm

Pseudo code of parallel merging algorithms for final clusters

```
1   RI – Relative inter connectivity
2   RC – Relative Closeness
3   α - user defined parameter
4   β – RI x RC
5   th – threshold value to take merging decision
6   n be number of clusters to be merge
7   Algorithm :
8   for i=0 ... n // i and j are used for clusters
        a.  for j=i+1 ... n
                i.  Assign task to work pool
                    merge(i,j);
               ii.  End for // iteration j
```

4. EXPERIMENTAL WORK AND RESEARCH

Advantage of chameleon algorithm is that it can be easily used to cluster two dimensional as well as three Dimensional data sets. With regard to parallel chameleon the Experimental work is carried out and the results are found improving the performance as compared to the serial chameleon. This paper compared parallel implementation with the serial chameleon algorithm with two different data sets of 8,000 and 10,000 data points. Parallel chameleon results in the same clustering as the serial implementation but it provides a better performance on the processors with multi-core architecture. Parallel chameleon is analyzed on two different processors Intel core i5 and Intel core i7.The algorithm performed a clustering method by distributing the points equally among threads so as to balance the results.

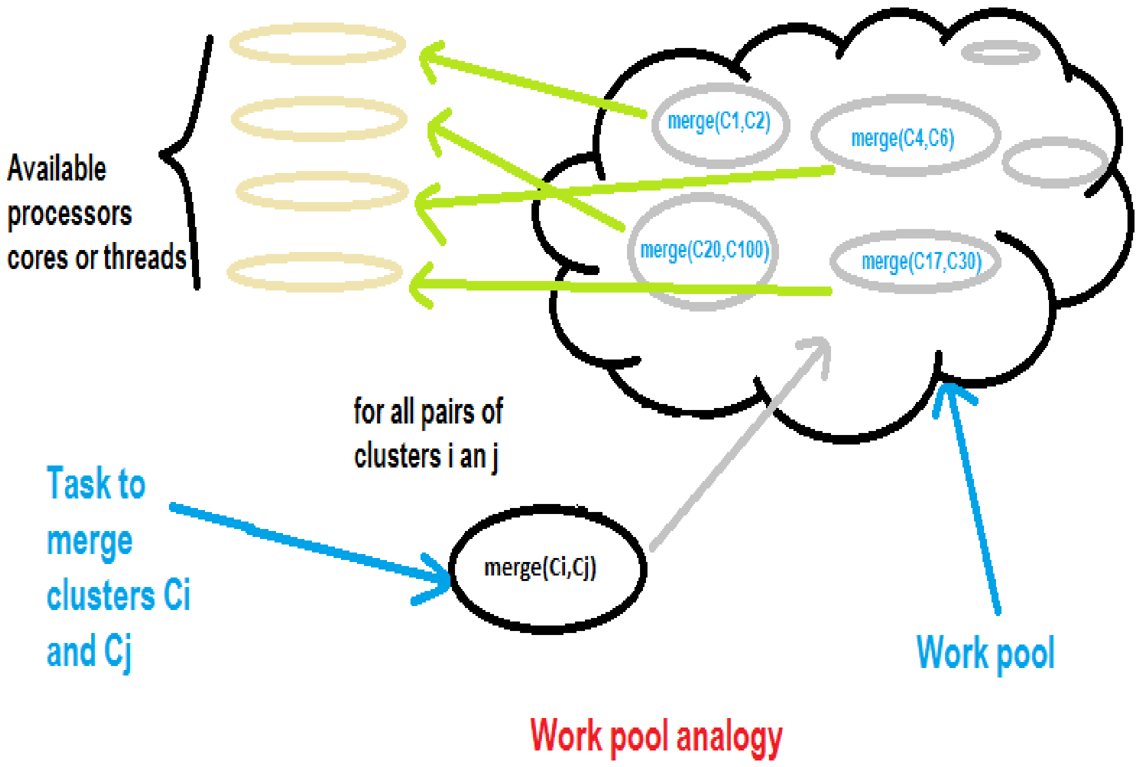
Available processors cores or threads
merge(C1,C2)
merge(C4,C6)
merge(C20,C100)
merge(C17,C30)
for all pairs of clusters i an j
Task to merge clusters Ci and Cj
merge(Ci,Cj)
Work pool
Work pool analogy

Fig2: **Work pool analogy for parallel chameleon.**

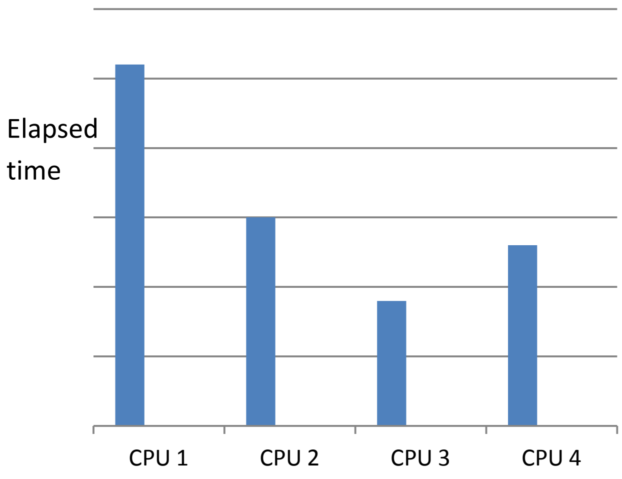

Fig 3: Simultaneously utilized CPUs.

The results obtained by each thread was combined to get the whole clustering output of the given data points as shown in Figure.The large set of data points were given as input to check the scalability of the algorithm.

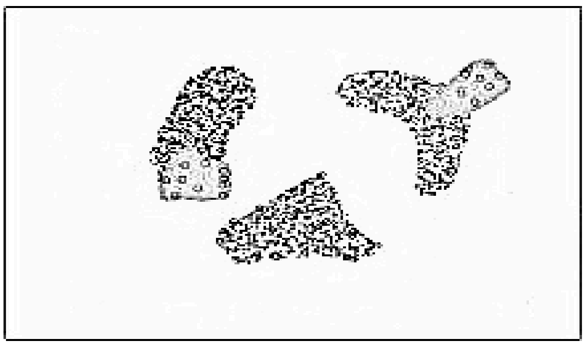

Fig 4: Data set with 8000 data points (C1).

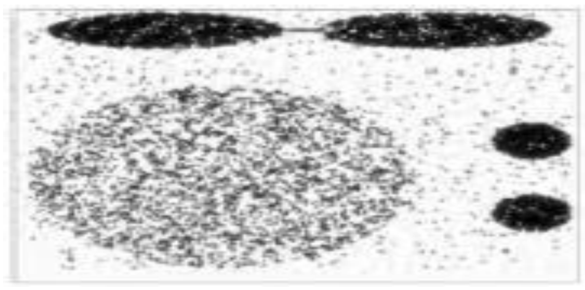

Fig 5: Data set with 10000 data points (C2).

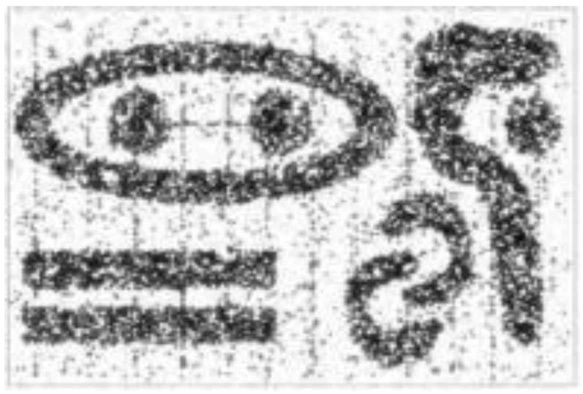

Fig 6: Data set with 20000 data points (C3).

Table 1: Summary of Performance analysis of parallel chameleon.

Processor	Number of threads	Number of Data Points	Speed Up as compared to single core processor
Intel core i5 processor	**4**	**8000**	**2.0**
Intel core i7 processor	**8**	**8000**	**4.5**
Intel core i5 processor	**4**	**10000**	**2.6**
Intel core i7 processor	**8**	**10000**	**4.8**
Intel core i5 processor	**4**	**20000**	**3.1**
Intel core i7 processor	**8**	**20000**	**5.5**

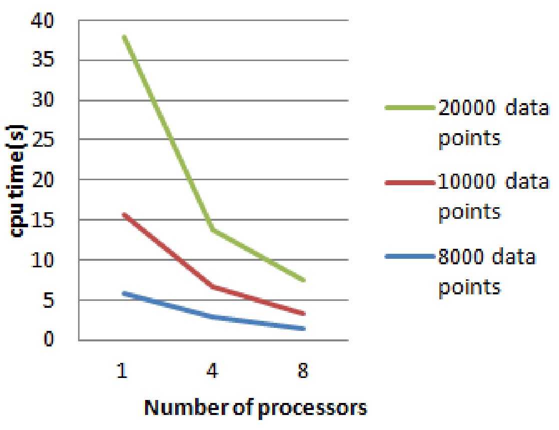

Fig 7: Comparative analysis on different processors for the three data sets.

For the set C1 with 8,000 data points get a speed up of approximately 4.46 on multi-core processors analyzed as compared to single core processor like Intel Pentium IV. While for the set C2 with 10,000 data points we get a speed up of approximately 2.6x on core i5 processor(with 4 threads) while on core i7 processor(with 8 threads) the speed up is near 4.8x as compared to single core processor like Intel Pentium IV. Also for the set C3 with 20,000 data points we get a speed up of approximately 2.8x on core i5 processor(with 4 threads) while on core i7 processor(with 8 threads) the speed up is near 5.0x as compared to single core processor like Intel Pentium IV.

5. CONCLUSIONS AND FUTURE WORKS

The parallel algorithm helps computing the data with a low time complexity independent of size of dataset. It works on symmetric and asymmetric multiprocessors. As openMP handles the asymmetric loop level nested parallelism, the algorithm gives performance gain over small scale and large scale SMPs. Then the portioning of data, thus, is highly reliable and parallel with the support of the in-built library. The task construct also helps to do a set of computations to be easily divided to threads this improves performance of functions in the program.

Future work includes a tool to perform dynamic modelling of hierarchical clustering in parallel .The algorithm will be thread-independent so it will make it viable to run across platform. The algorithm will be made in such a way that it works both on parallel and distributed systems.

6. ACKNOWLEDGEMENTS

Our sincere thanks to all the faculty members, acquaintances and friends who helped us bring this work across.

7. REFERENCES

[1] Hadjidoukas, P. E. & Amsaleg, L. Parallelization of a Hierarchical Data Clustering Algorithm Using OpenMP In Proc. the 2nd International Workshop on OpenMP (IWOMP ’06, 2006)

[2] Garcia, V.; Debreuve, E. & Barlaud, M. Fast k-nearestneighbor search using GPU *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on,* 2008, 1-6

[3] Karypis, G.; Han, E.-H. (S. & Kumar, V. Chameleon: Hierarchical Clustering Using Dynamic Modeling Computer, IEEE Computer Society Press, 1999, 32, 68-75

[4] Sismanis, N.; Pitsianis, N. & Sun, X. Parallel search of k-nearest neighbors with synchronous operations.*High Performance Extreme Computing (HPEC), 2012 IEEE Conference on,* 2012, 1-6

[5] Xu, R. & Wunsch D., I. Survey of clustering algorithms Neural Networks, IEEE Transactions on, 2005, 16, 645-678

[6] Maitrey, S.; Jha, C. K.; Gupta, R. & Singh, J. Article: Enhancement of CURE Clustering Technique in Data Mining.*IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues (DRISTI 2012),* 2012, DRISTI, 7-11

[7] J. Han and M. Kamber, “Data Mining: Concepts and Techniques”, Morgan Kaufmann.2000

[8] Karypis, G. & Kumar, V. Parallel Multilevel Graph Partitioning *Proceedings of the 10th International Parallel Processing Symposium, IEEE Computer Society,* 1996, 314-319

[9] Graph Partitioning Algorithms for Distributing Workloads of Parallel Computations (generals exam). Bradford L. Chamberlain. University of Washington Technical Report UW-CSE-98-10-03, October 1998.

[10] Foti, D.; Lipari, D.; Pizzuti, C. & Talia, D. Scalable Parallel Clustering for Data Mining on Multicomputers Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing, Springer-Verlag, 2000, 390-398.

[11] K. P. Soman, Shyam Diwakar, V. Ajay, InsightInto Data Mining: Theory and Practice, PHI Learning Pvt Ltd, 2006.

[12] Xu, X.; Jäger, J. & Kriegel; H.-P.A Fast Parallel Clustering Algorithm for Large Spatial Databases Data Min. Knowl. Discov., Kluwer Academic Publishers, 1999, 3, 263-290.

[13] George Karypis and Vipin Kumar A Hypergraph Partitioning Package Version 1.5.3. Army HPC Research Center. November 22, 1998

[14] https://developer.nvidia.com/cublas

[15] Guha, S.; Rastogi, R. & Shim, K. CURE: an efficient clustering algorithm for large databases Proceedings of the 1998 ACM SIGMOD international conference on Management of data, ACM, 1