Asynchronous Programming for Big Data using K-Mean Clustering

Md. Mizanur Rahman, Ohidujjaman, Mohammad Nurul Huda

Abstract— Clustering is very problematic on big data set in synchronous programming as well as time consuming. Finding optimal result from big misaligned data set is precarious because lack of seamless data pattern. This study emphasizes on reducing time fact and deployment of asynchronous program rather than synchronous model on big data. However, we provide an easy and flexible model named UACM (Unified Asynchronous Clustering Model) for clustering on multi dimensional, misaligned and big data set. The UACM replica segments big data set for parallel processing depending on segmentation of microprocessor. This model uses small unit of process (thread) as per the data set. The UACM architecture is also cost minimizing regarding hardware. This model evaluates the time fact between synchronous and asynchronous programming on large and misaligned data set. However, we implement the UACM model in real-time platform without help of clustering tools.

Index Terms— Asynchronous, Big Data, Cluster, Misaligned Data, Parallel Processing, Synchronous, Thread, UACM

1 Introduction

Clustering big data set is very essential for our real life. Data classification takes place a vital role for accessing and retrieving optimal result from big amount of data set. However, from the prior research we notice that there is very poor outcome for clustering big and misaligned data set [1] [2] [4]. The main reasons are processing technique and the difficulties of data set pattern. Most of the study predicts result on small amount of data set and uses synchronous processing. Nevertheless, most of the study avoided misaligned data set for clustering [3] [6]. The perception of the UACM model is clustering to big amount of data and misaligned data for retrieving optimal result.

We propose our model that utilizes the technique of asynchronous programming. The technique segments the microprocessor into optimal core segment. The big amount of data sets is equally divided depending on each core segment. The gain of this method is to reduce time fact of parallel processing at each core. Each of the execution is done on thread instead of process. This model in fact reduces average waiting time, turnaround time, response time and increase CPU utilization and throughput [12]. However, we finally can find the clustering result within less time for using this model. There are many clustering algorithms such as K-mean, Hierarchical, Fuzzy C-mean, Naïve Bayesian, Hidden Markov Model (HMM), and Viterbi etc. In this study, we consider the k-Mean algorithm in our experimental part.

We use big amount of data set with D dimension in our study. The clustering integers K are fixed previously set for each segment of data set.

1.1 Objectives

- ✓ Time reducing
- ✓ Big data clustering using limited hardware resources
- ✓ Big data clustering rapidly in real time
- ✓ Asynchronous programming utilization
- ✓ Microprocessor segmentation into **N** core
- ✓ Thread concept instead of processes
- ✓ CPU maximum utilization
- ✓ Maximum throughput
- ✓ Minimize turnaround time(TAT)
- ✓ Minimize response time (RT)
- ✓ Minimize waiting Time

1.2 Application

Clustering algorithms can be applied in many arena of real life. The proposed model (UACM) can be applied to cluster big data such as medical informatics, geographical analysis, customers' category analysis in financial organization such as banking sector, mobile operator, image processing and segmentation and so on [1] [7] [8][17]. The real life application fields of our proposed model are as follows:

- ✓ To cluster huge data in real time
- ✓ Stock market to categorize the customers
- ✓ To category the Data in search engine application
- ✓ In the Customers' data in banking sector
- ✓ In social market for grouping different community
- ✓ In the international aid organization
- ✓ To cluster the traditional culture in different community over the world
- ✓ Diseases' symptom clustering in medical sciences
- ✓ Product clustering in virtual markets

2 Literature Review

The existing scenarios are not well set up for reducing the time fact and compatibility of hardware. Most of the model follows the specific clustering algorithm in several cases. Conversely, our proposed model is uniform for all types of clustering algo-

- *Md. Mizanur Rahman is currently pursuing masters degree program in computer science and engineering in United International University, Bangladesh, PH: +88-01553507265. E-mail:rmic915@gmail.com*
- *Mr. Ohidujjaman is currently engaging as a lecturer of computer science and engineering in Daffodil International University, Bangladesh, PH:+88-01703991741. E-mail: jaman.cse@diu.edu.bd*
- *Mohammad Nurul Huda is currently engaging as a professor of computer science and engineering in United International University, Bangladesh, PH: +88-01673476433. E-mail: mnh@cse.uiu.ac.bd*

rithm. There is no model, which discusses simultaneously big amount of data, each data of multidimensional, maximum amount of cluster number and, mixed dimension of data set in parallel processing. The UACM is the only uniform unique model that converses all the above criteria. In mention that, there are few existing researches, which are partially little bit similar to our proposed model (UACM).

Open Multi-Processing (OpenMP) is shared memory architecture, which provides a multithreaded competence to the multiprocessor CPUs. Threads can share data from local shared memory during iterations of loop [3]. The basic idea behind OpenMP is SIMD for data-shared parallel execution [2]. The lack of OpenMP is to use only the homogeneous dimensional data and absence of asynchronous processing.

Compute Unified Device Architecture (CUDA) is an extension to simple programming languages, based on a few corresponding additions to programming language syntax [2]. Data transfer is asynchronous and takes place concurrently with several memory copies [9]. The problem of CUDA is that the data pattern not segmented with the creation of each thread. It accesses the data from the memory in random cases by specific formulation. The CUDA SDK (Software Development Kit) comes with tools that fully integrate with various C++ compilers [2]. That means CUDA acts with the help of specific software tools, which leads to the lack of dynamic programming.

Li and Fang [5] presented two parallel clustering algorithms on a single instruction multiple data (SIMD) architecture [4]. Kittisak Kerdprasop and Nittaya Kerdprasop [6] proposed the parallelization of the well-known k-means clustering algorithm by employing a single program multiple data (SPMD) approach based on a message passing model [4]. These two articles perform on fixed dimensional data set, limited clustering number that leads to lack of perfection in clustering.

Weerasak Chongnguluam et al.,[4] presented a article "Parallelized Rough K-means clustering with Erlang programming" that proposed implementation of the rough k-means clustering algorithm in parallel by using Erlang Programming. This research is depended one specific programming part in prediction optimal results. In the UACM architecture there is no specific bounding for applying any other programming. It always generates approximately suitable outcome.

3 Methodology

Big data clustering can be performed in different ways using many types of algorithm. However, from the existing sceneries we observe lots of clustering process which are very time consuming and poor performance of CPU utilization[2], [4]. There are many studies lunched to execute clustering big data in real life such as synchronous process, asynchronous process, parallel process and serial process etc[2] [5][6] [11] . We propose a very flexible Unified Asynchronous Clustering Model (UACM), which is assembled by asynchronous programming with threads.

3.1 Synchronous Programming

Synchronous programming is a means of sequential programming [10] [11]. The execution happens in a single series in Synchronous processing such as X_1--->X_2----->X_3---->…....-------->Xn. If CPU allocates X_1, rest of the jobs will wait until to complete the X_1. After completion of X_1 CPU further allocates next one from the ready queue and so on [12]. During the allocation of CPU for processes is called the CPU Scheduling Time (CST) of each process. For the period of each process termination from CPU, is called the process Completion Time (CT). The process view of synchronous processing is shown in fig 1. Synchronous processing of N routines are X_1, X_2, X_3…....Xn:

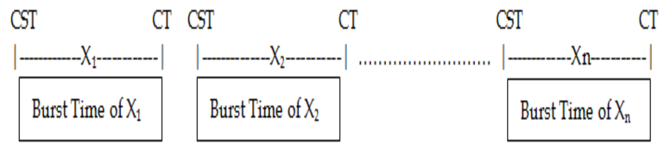


Fig. 1. Schematic diagram of synchronous processing

3.2 Asynchronous Programming

Asynchronous programming is a means of parallel programming [13] [14]. It operates with random execution of multiple processes. The job can be processed separately using multithread of each core. The CPU can allocate a new job though the first one still is not finished and so on. The concept is that CPU can allocate X_3 while X_1 and, X_2 are still running. Nevertheless, this concept leads to save time and increase maximum number of instances of a resource [12]. The asynchronous process view is shown in the Fig 2 Asynchronous processing of N routines are X_1, X_2, X_3…...................…Xn:

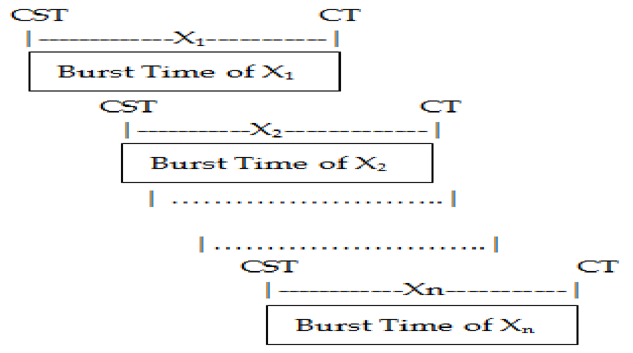


Fig. 2. Schematic diagram of Asynchronous processing

3.3 Proposed Model-UACM (Unified Asynchronous Clustering Model)

There are lots of clustering algorithms such as k-mean, modified k-Mean, Fuzzy c-Mean, Hierarchical Clustering, Nearest Neighbor, Naïve Bayesian, and SOM etc. The uniform flowchart of unified asynchronous clustering model (UACM) is as follows:

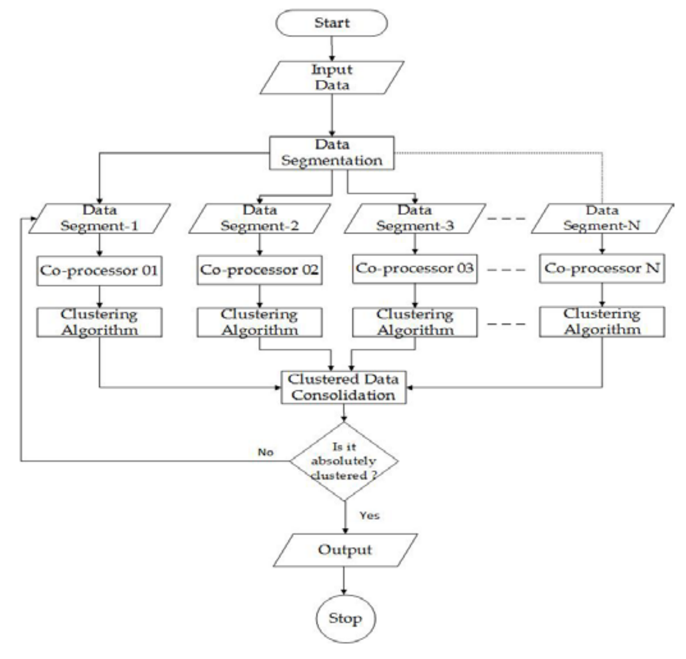


Fig. 3. Flowchart of Unified Asynchronous Clustering Model (UACM)

The uniform flowchart of unified asynchronous clustering model (UACM) is for every types of clustering algorithms. This is absolute dynamic model for clustering big data in asynchronous pattern.

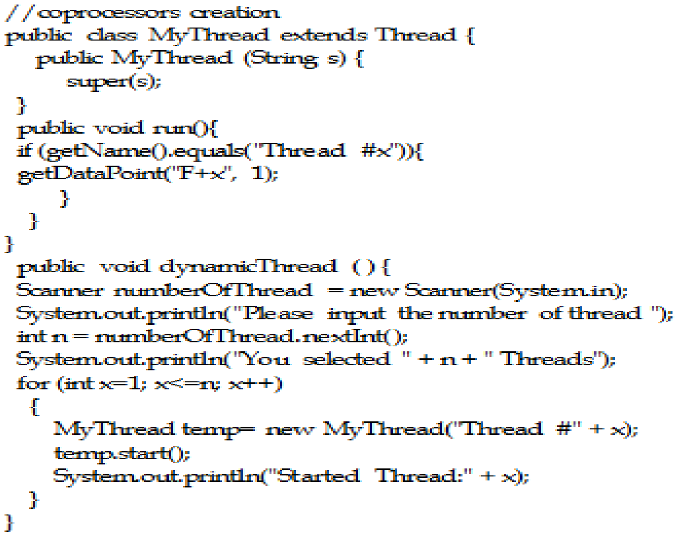
The following code segment is for creation the n number of data segments.

```
//Data Segmentation
public class DataSegment{
 public void ClusterData( int segmentNo) {
 int segmentFileSize = 0;
 int startPoint = 0;
 for (int i = 0; i < segmentNo; i++) {
   try {
     FileInputStream input = new FileInputStream("D:\\F.txt");
      try {
        FileOutputStream output = new
        FileOutputStream("D:\\F" + (i+1)+".txt");
         try {
          byte[] bytearr = new byte[9000000];
          int filesize = input.read(bytearr);
           if (segmentFileSize != 0) {
             startPoint = startPoint + segmentFileSize;
             }
             segmentFileSize = filesize/ segmentNo + 1;
             output.write(bytearr, startPoint,segmentFileSize);
             UpdateSize=segmentFileSize+UdateSize;
             fileSize=FileSize-UpdateSize;
             if(segmentFileSize> fileSize){
             filesize = input.read(bytearr);
             segmentFileSize=filesize % segmentFileSize;
             output.write(bytearr, startPoint,segmentFileSize);
             }
             filesize = input.read(bytearr);
            } finally {
               output.close();
              }

          } finally {
            input.close();
            }

        } catch (IOException e) {
             e.printStackTrace();
          }
      }
   }
}
```

The following code segment is for creation the **n** number of coprocessors.



```
//coprocessors creation
public class MyThread extends Thread {
   public MyThread (String s) {
      super(s);
 }
 public void run(){
 if (getName().equals("Thread #x")){
 getDataPoint("F+x", 1);
     }
  }
}
 public void dynamicThread (){
 Scanner numberOfThread = new Scanner(System.in);
 System.out.println("Please input the number of thread ");
 int n = numberOfThread.nextInt();
 System.out.println("You selected " + n + " Threads");
 for (int x=1; x<=n; x++)
  {
    MyThread temp= new MyThread("Thread #" + x);
    temp.start();
    System.out.println("Started Thread:" + x);
  }
}
```

We have chosen k-Mean algorithm in our experimental part. Pseudo code of the k-Mean clustering algorithm is as follows:

```
Step -1: Cluster:=K , Dimension:=D
         Number_of_Centroid:=Number_of_cluster
         Initial Centroid:=
         $(X_1,Y_1,- -,D_1)$, $(X_2,Y_2,- -,D_2)$,- - - -,$(X_k,Y_k,- -,D_k)$
Step-2: Generate k cluster from given data. Assign each data
         into nearest cluster.
Step-3: Recalculate new cluster centroid. Check the new cen
         troid with immediate prior centroid.
Step-4: Repeat step 2 and step 3 until to find the exact similar
         centroid with immediate prior centroid.
```

The unified asynchronous clustering model architecture regarding k-Mean algorithm is below:

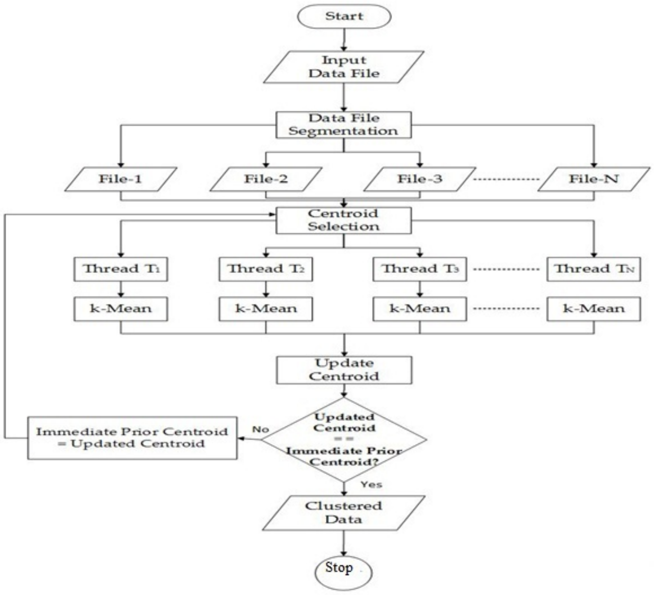


Fig. 4. Architecture of Unified Asynchronous Clustering Model (UACM)

The unified asynchronous clustering model (UACM) based on k-Mean clustering algorithm. It is perceivable that large amount data file is segmented into n numbers of file. According to the numbers of segment file, equal numbers of thread are generated. Each of the thread is utilized using k-Mean clustering algorithm, which leads to asynchronous programming [14] [15].

4 Result Discussion

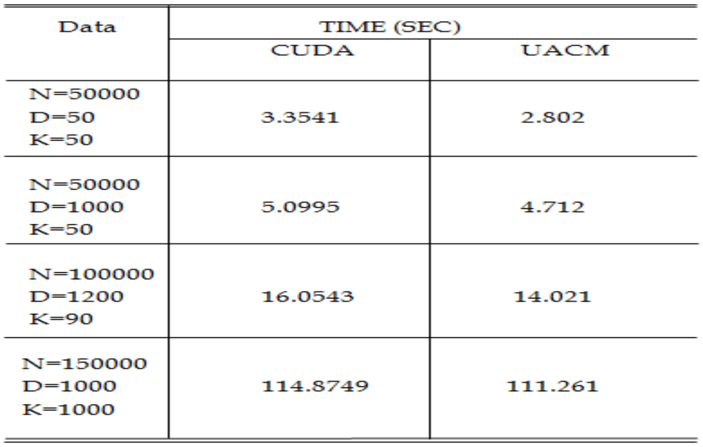
The experimental part is performed by the Dell computer, (model PowerEdge T320 Tower Server) which is configured by 2.4 GHz Intel Xeon processor E5-1410, 16 GB RAM, external graphics memory card.

The raw data are acquired by applying IBM Quest Synthetic Data Generator tool [2][18]. After accomplishing the data we preprocess these data into a file that is assumed as the back ground of database. The experimental part is performed on k-Mean clustering algorithm regarding UACM architecture. In our study, we use the object oriented programming language such as java.

In our experiment we perform to produce different result by considering four level of criteria such as number of data N=50000, dimension D=50 and cluster k=50. The second, third and fourth level criteria are number of data N=50000, dimension D=1000 and cluster k=50, number of data N=100000, dimension D=1200 and cluster k=90 and, number of data N=150000, dimension D=1000 and cluster k=1000 respectively [2]. The acquired result of the experimental part is shown comparing with some existing model such as Open Multi-Processing (OpenMP) , Compute Unified Device Architecture (CUDA) , Hybrid (Hybrid=CUDA + OpenMP) model. The table 1 shows the time comparison between CUDA and UACM. It is perceivable that each of the outcome result is better in UACM model [2] [4] [16].

Time comparison result between OpenMP and UACM model

TABLE 1
THE TIME COMPARISON BETWEEN CUDA AND UACM



Data	TIME (SEC)	
	CUDA	UACM

N=50000
D=50
K=50
	3.3541	2.802

N=50000
D=1000
K=50
	5.0995	4.712

N=100000
D=1200
K=90
	16.0543	14.021

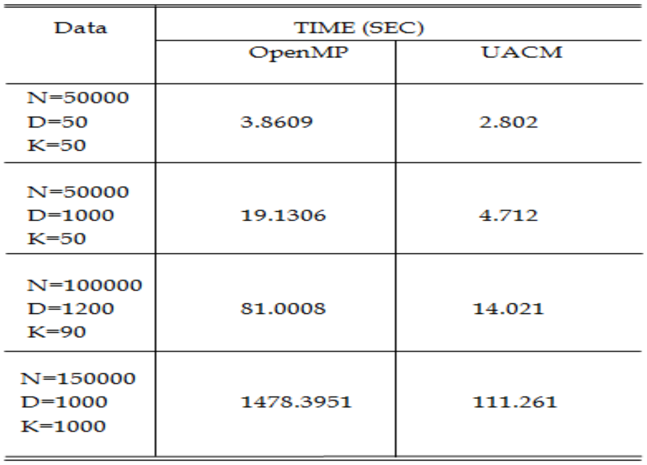
N=150000
D=1000
K=1000
	114.8749	111.261

N= Number of data point, D= Number of data dimension, K=number of cluster, CUDA=compute unified device architecture, UACM=.unified asynchronous clustering model.

is shown in table 2. It is noticeable that increasing of dimension and cluster number, the time unit is rapidly rising in OpenMP but slightly in UACM model [2][4].

However, Table 3 shows time comparison between Hybrid and UACM. It is observable that the time unit is increasing approximately likeness between Hybrid and UACM in case of rising data set, number of dimension and cluster number [2][4][16].

TABLE 2
THE TIME COMPARISON BETWEEN OPENMP AND UACM



Data	TIME (SEC)	
	OpenMP	UACM

N=50000
D=50
K=50
	3.8609	2.802

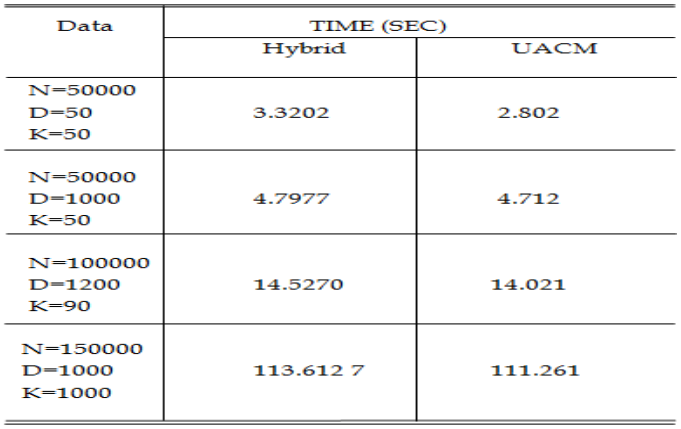
N=50000
D=1000
K=50
	19.1306	4.712

N=100000
D=1200
K=90
	81.0008	14.021

N=150000
D=1000
K=1000
	1478.3951	111.261

N= Number of data point, D= Number of data dimension, K=number of cluster, OpenMP=Open Multi-Processing, UACM=.unified asynchronous clustering model.

TABLE 3
THE TIME COMPARISON BETWEEN HYBRID AND UACM



Data	TIME (SEC)	
	Hybrid	UACM

N=50000
D=50
K=50
	3.3202	2.802

N=50000
D=1000
K=50
	4.7977	4.712

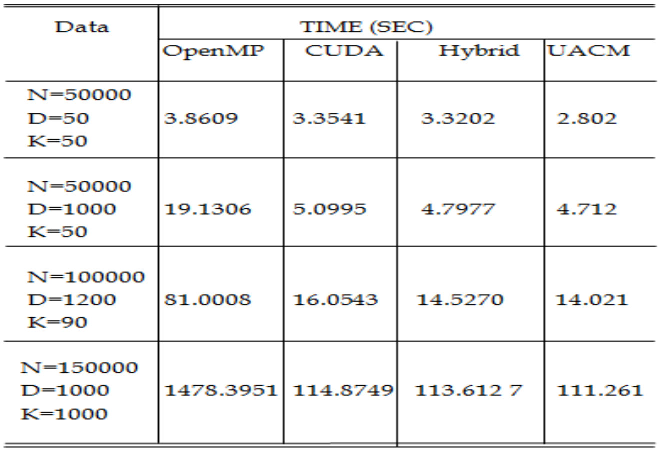
N=100000
D=1200
K=90
	14.5270	14.021

N=150000
D=1000
K=1000
	113.612 7	111.261

N= Number of data point, D= Number of data dimension, K=number of cluster, Hybrid= CUDA+OpenMP, UACM=.unified asynchronous clustering model.

Table 4 shows the comparison among various implementations. In such view, OpenMP model gives very worst experimental results. CUDA and Hybrid models produce about similar result [2]. Nevertheless, the UACM model provides us the best and significant outcome.

TABLE 4
THE TIME COMPARISON AMONG OPENMP, CUDA, HYBRID AND UACM



Data	TIME (SEC)			
	OpenMP	CUDA	Hybrid	UACM

N=50000
D=50
K=50
	3.8609	3.3541	3.3202	2.802

N=50000
D=1000
K=50
	19.1306	5.0995	4.7977	4.712

N=100000
D=1200
K=90
	81.0008	16.0543	14.5270	14.021

N=150000
D=1000
K=1000
	1478.3951	114.8749	113.612 7	111.261

N= Number of data point, D= Number of data dimension, K=number of cluster, OpenMP=Open Multi-Processing, CUDA=compute unified device architecture, Hybrid= CUDA+OpenMP, UACM= unified asynchronous clustering model.

The fig 5 shows the statistical distinguish result pattern of four levels of criteria **(N=50000, D=50, k=50; N=50000, D=1000, k=50; N=100000, D=1200, k=90 and N=150000, D=1000, k=1000 and represented by 1 ,2 3 and 4 respectively in the fig 5)** regarding OpenMP, CUDA, Hybrid and UACM methods. The time value is required maximum for large amount of data, dimension and cluster in OpenMP model. Nonetheless, limited time unit is required for our proposed model (UACM) though the amount of data, dimension and cluster are in large scale[2][4][16].

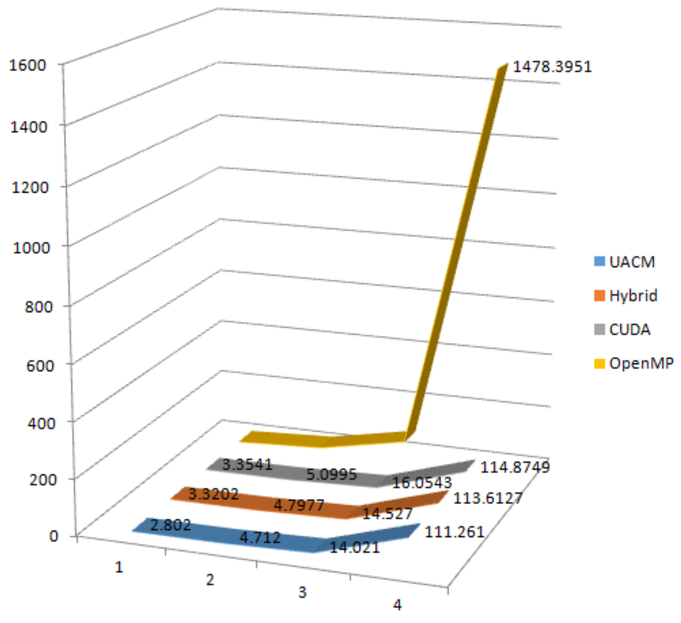


Fig. 5. Statistical result analysis amongst OpenMP, CUDA, Hybrid and UACM methods.

5 Conclusion

In our study, we use a unique model (UACM) for clustering big amount of data regarding all kinds of clustering algorithms. In this research, the experimental results are acquired by designing raw code depending on the unified asynchronous clustering model. There is no use of clustering tools software like Weka , Clementine etc in the empirical part. The significant focus of this research is to classify the mixed and multi-dimensional large amount of data in parallel processing system.

The asynchronous processing gives us the tremendous opportunity to use of microprocessor in N segment. Ultimately, data processing on big facts are not time consuming in asynchronous processing rather than synchronous.

REFERENCES

[1] Fazilah Othman et al., "Parallel K-Means Clustering Algorithm on DNA Dataset" School of Computer Science, Universiti Sains Malaysia, 11800, Penang, Malaysia {fazot, rosni, nuraini, rosalina}@cs.usm.my

[2] Ghudji R. and Padmavathi B., " K-Means for Parallel Architectures", International Journal of Advanced Research in Computer Science and Software Engineering 5(2), February - 2015, pp. 256-262

[3] C. Yang, C. Huang and C. Lin,"Hybrid CUDA, OpenMP and MPI parallel peogramming on multicore GPU," ELSEVIER transaction in Computer Physics Communications 182 (2011) 266-269.

[4] Weerasak Chongnguluam et al., "Parallelized Rough K-means clustering with Erlang programming" Proceedings of the International MultiConference of Engineers and Computer Scientists 2011 Vol II, IMECS 2011 ,March 16-18, 2011, Hong Kong

[5] X. Li and Z. Fang, "Parallel clustering algorithms", Parallel Computing, Vol.11, Issue 3, 1989, pp. 275-290.

[6] K. Kerdprasop and N. Kerdprasop, "Parallelization of K-means Clustering on Multi-core Processors", in Proceedings of 10th WSEAS International Conference on Applied Computer Science, Japan, October 2010, pp.472-477.

[7] Cristia n J., Lizeth J., and Luciano A. R., "Parallelization of the Algorithm k-means Applied in Image Segmentation " , International Journal of Computer Applications (0975 8887) . (2014)

[8] Xiufeng Wan et al., "Interactive Clustering for Exploration of Genomic Data", Mississippi State University, Mississippi State, MS USA,2002

[9] Liheng Jian et al., "Parallel data mining techniques on Graphics Processing Unit with Compute Unified Device Architecture(CUDA) ", The Journal of Supercomputing Springer , Volume 64, Issue 3, June 2013.

[10] Ashish A., Shailendra and Shende W., " IMPLEMENTATION OF K-MEANS ALGORITHM BY USING MAP REDUCTION AND BULK SYNCHRONOUS PARALLELISM" International Journal of Electrical, Electronics and Computer Systems (IJEECS), ISSN (Online): 2347-2820, Volume -2, Issue-1, January, 2014

[11] Klaus Schneider , "The Synchronous Programming Language Quartz" A Model-Based Approach to the Synthesis of Hardware-Software Systems, Department of Computer Science University of Kaiserslautern, November 13, 2010, Version 2.0

[12] Schilberschatz A., Peter B. and Gange G., "Operating System Concepts" book, Publisher: John Wiley & Sons Inc 9th edition

[13] Thomas Lieber, "Understanding Asynchronous Code" Submitted to the Department of Electrical Engineering and Computer Science on May 22, 2013, in partial fulfillment of the requirements for the degree of Master of Science

[14] Roy Krischer, "Advanced Concepts in Asynchronous Exception Handling" A thesis presented to the University of Waterloo in fulfillment of the thesis requirement for the degree of Doctor of Philosophy In Computer Science Waterloo, Ontario, Canada, 2010

[15] Andrew D. et al, "An introduction to programming with threads. Technical Report Digital Systems Research Center, 130 Lytton Avenue, Palo Alto, California, 94301, January 1989. 15, 93

[16] Kerdprasop K. and Kerdprasop N., "Parallelization of K-Means Clustering on Multi-Core Processors" Data Engineering and Knowledge Discovery (DEKD) Research Unit School of Computer Engineering, Suranaree University of Technology 111 University Avenue, Muang District, Nakhon Ratchasima 30000, THAILAND

[17] Jing Zhang et al., "A Parallel Clustering Algorithm with MPI – MKmeans" JOURNAL OF COMPUTERS, VOL. 8, NO. 1, JANUARY 2013

[18] "IBM Quest Synthetic Data Generator Tool" https://sourceforge.net/projects/ibmquestdatagen/

[19] T. Sajana, C. M. Sheela Rani and K. V. Narayana, "A Survey on Clustering Techniques for Big Data Mining" Indian Journal of Science and Technology, Vol 9(3), DOI: 10.17485/ijst/2016/v9i3/75971, January 2016

[20] Mugdha Jain M. , Verma C., "Adapting k-means for Clustering in Big Data" International Journal of Computer Applications (0975 – 8887) Volume 101– No.1, September 2014

[21] Kaur Toor A. and Singh* A., "An Advanced Clustering Algorithm (ACA) for Clustering Large Data Set to Achieve High Dimensionality", J Comput Sci Syst Biol Volume 7(4)115-118 (2014) – 0118, ISSN: 0974-7230 JCSB, an open access journal