Volume 2 No. 4, April 2012 ISSN 2223-4985

International Journal of Information and Communication Technology Research

http://www.esjournals.org

Performance Analysis of High Performance k-Mean Data Mining Algorithm for Multicore Heterogeneous Compute Cluster

Ramesh Singh Yadava[1], P.K.Mishra[2]
[1]Computer Center,
Banaras Hindu University,Varanasi-221005
[2]Department of Computer Science,
Banaras Hindu University,Varanasi-221005

ABSTRACT

In this paper, we have study the performance of k-Mean data-mining algorithm (k-Mean),which is implemented on the heterogeneous compute cluster with the multi core programming. The multicore program is implemented with MPI and C for the parallel computing and utilizing the maximum compute power of the heterogeneous cluster. The heterogeneous cluster is established with the help of MPICH2.

We have also analyzed the efficiency and performance of k-Mean data mining algorithm for the large dataset. The dataset, which we have used, is chess.txt [1]. The dataset is divided into the number of cores and core compute the dataset independently and makes a data cluster of similar dataset on each processor core.

Through this implementation, we have justified that the communication time among the processor cannot be negligible for large dataset. So, the compute time for same dataset on different processors core with same speed and memory is different and the different processor with different speeds and memory access also take different time.

Keywords: *MPI (Message Passing Interface), MPICH2 (High performance & widely portable implementation of MPI)*

1. INTRODUCTION

Data mining is a process to discover similar data from the database/dataset in the different areas like finance, retail industry, science, statistics, medical sciences, artificial intelligence, neuroscience, etc [2]. The data sizes are increasing very fast, so, new and high-speed technologies and algorithms are needed for collecting and processing the data. The parallel data mining algorithms are required for data mining the data cluster in high-performance computing. For theses purpose some algorithms has been implemented in the Cluster Computing environment such as k-Mean, BIRCH (balance iterative reducing and clustering using hierarchies) [3, 4]. But none of the data-mining algorithm has been implemented for the multicore programming for parallel processing environment on the different dataset.

The parallel data mining algorithms are required Cluster of the multicore processor to process the dataset or database.

In the past few decades, the data mining algorithms were developed for the parallel processing on the distributed or cluster or grid computing environments [5, 6]. Nowadays the algorithms are required to implement for the multicore to utilize the full computation power of the processors. In the modern era the 8 to 12 core processor with the shared cache and memory has been developed and deployed and in the future probably more core processor may be developed and deployed.

There are different types of Compute Clusters are available for the parallel computation. These compute cluster may be homogeneous or heterogeneous according to the compute machine used in the cluster. The compute cluster may be categorized on the basis of the machines used for the implementation of the compute cluster. The machine used for the development of the heterogeneous compute cluster may be with different compute power with different memory. The cluster may be 'Load Balancing Cluster' High Performance Cluster', High Availability, Cluster [7]. The high Performance/High availability cluster can be implemented with MICH2 and MAUI.

The MPICH2 cluster can be homogeneous or heterogeneous. It depends on the availability of the physical Compute Cluster. The MPICH2 is available for heterogeneous as well as for homogeneous cluster [8].

The MPI library may be used with C/C++/FORTRAN for developing the parallel programming. In the parallel programming the dataset/database and intermediate result of the particular processor may be communicated among the processor as and when require [9]. The data mining algorithms can be implemented through parallel programming for optimizing the use of the processor and improving the performance of the algorithms.

The k-Mean algorithm is one of the most using algorithms for data mining and data clustering [10]. This algorithm has been implemented for the parallel computation. The k-Mean algorithm can be developed for multi core for the parallel processing.

The k-Mean algorithm has been proposed for the implementation on Multi-core processor with the mapped memory to increase the computing performance. The advantage of this analogy is a large dataset may be executed on the Compute Cluster [11].

2. LITERATURES REVIEW

The k-Mean algorithm has been implemented for large dataset on the parallel computing. Since the large dataset cannot be executed on a single machine, due to the limitation of the memory. So, the k-Mean algorithm developed and implemented on the cluster of the Computers. For this purpose the dataset was divided into the number of the compute node, which are available in the compute cluster [12]. The data cluster of the similar data is created on each compute node and communicated to the other compute node, when it require. The similarity of the data point is measured with the Euclidean distance [12]. The mean distance is measured between the arbitrary point and all other points, and the data point, which distance is minimum, is stored in the appropriate data cluster accordingly [13, 14, 15].

The k-Mean algorithm is implemented for very large dataset on the available core of the compute machine with the shared memory. For this purpose the dataset is divided with the number of the processor cores, where the computing will be performed for data mining and data clustering. This is a parallel implementation of the k-Mean algorithms. This implementation was performed with the MPI and C. This implementation utilizes the maximum computing power of the processor and improves the performance of the k-Mean algorithm [16,17].

A non-hierarchical k/h k-Mean algorithm was developed for the numerical data. This algorithm was implemented on the cluster of 32 PCs. The dataset used for this purpose was a database. The database was distributed into the number of the PCs. The efficiency of the computation was improved in respect of previous parallel k-Mean algorithms. The algorithm was implemented with a pseudo code. In the pseudo code measuring, the computational time was a difficult problem on each compute node [18]. This implementation was not a multicore implementation to utilize the full computing power of the Compute Cluster.

The k-Mean and Apriori data mining algorithm was implemented with the GPUMiner system [19]. The GPUMiner system is a one of the most useful system for data mining and data clustering. The researchers have used three modules of the GPUMinner. The first module of the GPUMinner contain CPU based buffer for transferring the data between CPU and I/O and second module contain GPU-CPU based parallel co-processing unit and third module contain visualization based data mining module. The visualization module is used for implementing the data-mining algorithms, through which the user can get the graphical user interface (GUI). The data mining algorithms (k-Mean and Apriori) were implemented with GPUMinner, and the performances of these algorithms were improved significantly, and the computational speedup is also improved significantly. The API languages DirectX and OpenGL were used for these implementations [19].

The k-Mean data-mining algorithm was implemented in the parallel computing with the Erlang programming language. This programming language supports functions and message passing programming. The different functions may be used for different purposes. The k-Mean data-mining algorithm is implementation with this language and the performance of the algorithm is improved over the sequential k-Mean data-mining algorithm [20].

Another data mining algorithm, k-Mediod for data mining and data clustering was proposed and implemented sequentially and compared with DBSCAN data mining & data clustering algorithm and, it was observed that k-Mediod algorithm has given better performance than the DBSCAN data mining algorithm for data points scattered around some particular points [21]. Similarly another data mining & data clustering algorithm refinement k-Mean was proposed and implemented for the multidimensional binary search tree [22].

k-Mean algorithm is used for data clustering as well as data mining for making the cluster of the similar data point from a large dataset. So, this algorithm is also used for finding the similarity of the protein. The NCBI (National Center for Bio-Technology Information) protein has used this algorithm for this purpose. The Dataset used for finding the protein similarity was very large and that dataset is also increasing very fast. The sequential k-Mean was used for this purpose [23]. So, the parallel k-Mean algorithm is needed for protein data clustering and data mining.

In the recent years, the computer chip has been developed for collaborating with application program for the Multicore processor for parallel processing. This chip will improve the processor computing power and this computing power can be utilized for the implementation of the parallel data-mining algorithm, for improving the performance of the data mining and data clustering algorithms. The k-Mean algorithm was proposed for the multicore programming [24]. These implementations provide a new dimension for the parallel processing for the implementation of the data mining algorithms.

3. K- MEAN ALGORITHM FOR MULTICORE

We have implemented k-Mean data mining & clustering algorithm for multicore. The k-Mean algorithm is given below:

```
#include "mpi.h"
#include < stdio.h >
#include < stdlib.h >
#include < time.h >
#define SIZE_k 31960
#define SIZE_X 3196
#define SIZE_Y 40

1. Define all the global variables used on
   all processors.
2. Define the arbitrary data points for
   making the data cluster.
3. int main(int argc, char *argv[])
   Begin
4. Initialize:

   MPI program MPI_Int(&argc,&argv);
   Nodes MPI_Comm_Rank(MPI_COMM_WORD,&nodeid)
   Processor nos MPI_Comm_Rank(MPI_COMM_WORD,&np)

5. Store the text flat file in a matrix of $[mxn]$

6. Divide the matrix horizontally with no. available
   processor in the compute
   node $[increment = m / p]$

7. Initialize:

MPI_Send(& start_row,1,MPI_INT,i,41,MPI_COMM_WORLD);
MPI_Send(&end_row,1,MPI_INT,i,41,MPI_COMM_WORLD);


8. Start the time mesurment with MPI_Wtime();
   start_time= MPI_Wtime()

9. Receive the data from the sender node with

MPI_Recv(&start_row,0,MPI_INT,i,41,MPI_COMM_WORLD);
MPI_Recv(&end_row,0,MPI_INT,i,41,MPI_COMM_WORLD);

10. Calculate distance between the arbitrary data
    points and all other point for making the data
    cluster.
    Distance[j]=abs[n[i][1]-c_c[j])

11.Find minimum distance among all the distance and
   put the data point accordingly.

12.Comput the time take on the particular processor.
   End_time=MPI_Wtime();

   Time_taken=End_time-Start_time;

13.Repeat step (7) to (12) for each processor for
   computing the time.
14.The MPI_Finalize(); function is used in the
   last of all above processes.
```

The description of the k-Mean algorithm is given in the following:

Step I. The C and MPI libraries are included in the implementation and the variables are defined in (1) and arbitrary points are declared in (2).

Step II. The main function is defied in (3) and the MPI is initiated in (4) for the use of the MPI library functions.

Step III. The flat text file datasets are retrieved and converted into integer and stored in a matrix of $[mxn]$

Step IV. The matrix is divided into the number of processor core available in the compute cluster horizontally. Such as, let the numbers of processor are 'p', numbers of rows in the matrix are 'm' and number of rows processed on each compute core is '*increment*'. So, the number of rows given on each processor will be: $[increment = m/p]$ in (6).

Step V. The datasets are sent on the particular processor core with 'MPI_Send()' library function in(7) such as

```
MPI_Sendu&start_row,1,MPI_INT,i,41,MPI_COMM_WORL
D);
MPI_Send(&end_row,1,MPI_INT,i,41,MPI_COMM_WORLD)
;
```

In this 'start_row' is the beginning of the row number and '1' is the processor number, MPI_INT is initialization, 'i' is processor core number, 41 is the number of columns and MPI_COMM_WORLD is the MPI communication functions.

Similarly '$end\_row = start\_row + increment$' is the last row for sending the data to the compute processor core and other are same parameter as mentioned above. These two MPI_Send (…) statements will continue with the start_row and end_row swapped each others, till all the data is not sent to the available processing cores.

Step VI. The time of a particular processing core can be measured with MPI_Wtime() in (8) and

```
MPI_Recv(&start_row,0,MPI_INT,i,41,MPI_COMM_WORL
D);
MPI_Recv(&end_row,0,MPI_INT,i,41,MPI_COMM_WORLD)
;
```

in (9) ,received the datasets, which were send from the master node.
In the above MPI_Recv(…) function 'start_row' is the beginning of the row number and '0' is the processor number, MPI_INT is initialization, 'i' is processor core number, 41 is the number of columns and MPI_COMM_WORLD is the MPI communication functions.

Step VII. In the point (10) the Euclidean distance measurement function is used for making the data cluster of similar datasets. The similar dataset are made with measurement of shortest distance between the arbitrary points and data points in (11).The shortest distance data points are stored to data clusters accordingly.

Step VIII. Total time taken for the computation of the data on the particular core is measured in (12).

The Step VI to VIII has been repeated on all compute processor cores in (13) and in (14) MPI_Finalize(); function is used to show the end of the MPI program.

4. CLUSTER ARCHITECTURE

The above algorithm is implemented on the following heterogeneous Compute Cluster. The architecture of the cluster is given in following figure 1.

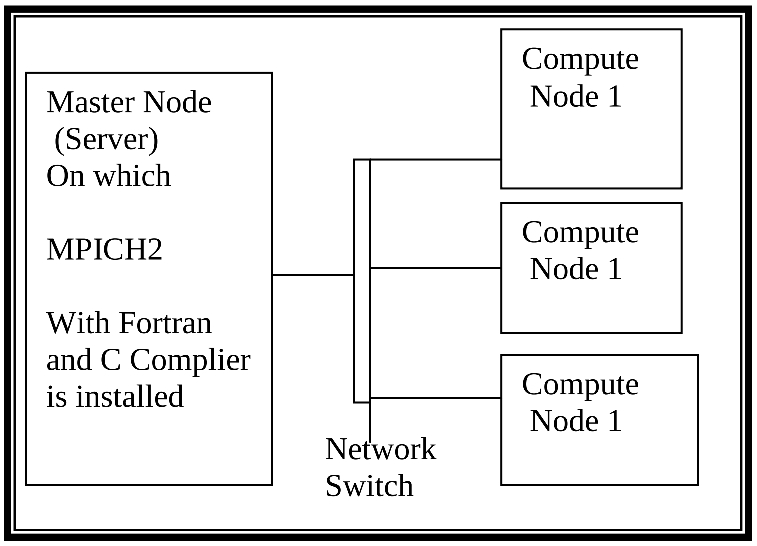

Figure-1 (Architecture of the cluster which we have used)

5. PERFORMANCE ANALYSIS

We have implemented the k-Mean data mining/clustering algorithm on MPICH2 Cluster. The MPICH2 Cluster, which we have made with four Computer (PC), in which two Computer (PC) are Core to Duo and two are Pentium 4.The Master node is core to Duo with Speed 2.80 GHz and two GB Ram and slave Computer (PC) is Core to duo with 2.80 GHz speed and 1.26 GB RAM and next two Computer (PC) are Pentium 4 with 1.60 GHz and one GB RAM. Our compute cluster is a heterogeneous cluster in two different ways. One is on the basis of the different computing speeds and second is on the different memory (RAM). These computers are connected with 100MB speed Ethernet switch.

The k-Mean data mining/clustering algorithm is implemented with MPI and C. This implementation is a multicore parallel implementation of the k-Mean algorithm. This program executed parallel on the available core of the cluster. The MPI_Send() and MPI_Recv() function is used for this purpose. For testing the performance of k-Mean data-mining algorithm, we have used chess.txt dataset, which is available on http://fimi.ua.ac.be/data/

The chess.txt is a dataset file. The content of this file is stored in a two dimensional array of size 3196x40 matrix. This matrix data is divided into five equal parts for parallel the computing on the five core of the heterogeneous cluster. The divided data size is 40x638 matrix for each processor. The processors (core) take the different times for the same dataset for making the cluster. The time taken for the core available on the master node is 1.5 second and the cores available on the next core to duo machine takes 11.4 second and 22.37 second and the Pentium processors takes 29.82 and 39.42 second respectively.

A graph is plotted between the number of processor and time for computing for the data cluster, is given following figure 2.

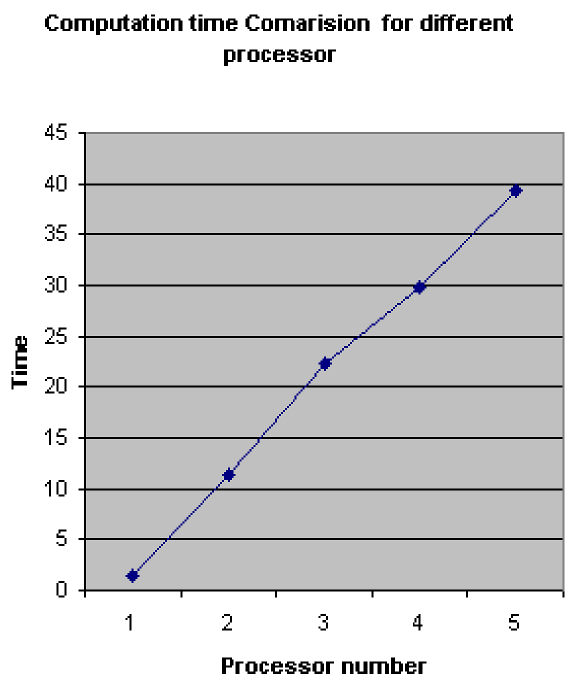

Figure-2 (Graph between number of processor and time taken in computation)

From the Figure-2, we find that the time is increasing as the speed and memory is decreasing and also so that the communication time can't be neglected for the large dataset.

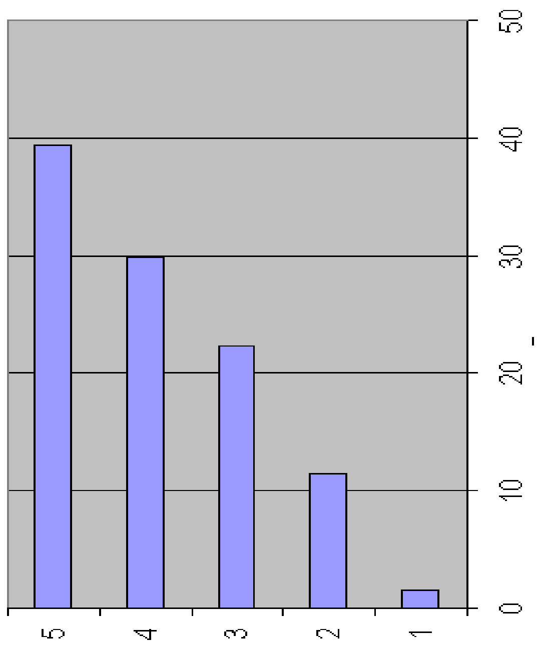

Figure-3 (Gant Chart for each processor for time taken on each processor for computing the same dataset)

We have also draw a bar chart between the number of processor and computation time and observe that the time is increasing, as the speed of the processor is decreasing and the communication time is also including in the computation time.

6. CONCLUSIONS

We have implemented parallel k-Mean algorithm on multicore and observe that the overall performance of the throughput time is improved but the time taken on different processor is different according to its speed and memory available for the cores. The communication time among the processor was also measurable. So, the computation time on the same speed and memory processor increased too. We also observe the limitation of the compute cluster with its memory. For executing large database, we require sufficient large memory otherwise the program can't be executed. For this purpose, we have used the mushroom dataset, whose size is very large in the comparison of the chess dataset. So, the mushroom dataset can't be executed on this Compute Cluster for data mining/clustering, since the memory size of this cluster is limited up to 2.5 GB after the running of the mpdboot and Operating System.

This implementation provide an idea for the future that a large dataset/database requires a high-performance and a large memory compute cluster for data mining/clustering, which can not be possible to implement on the single compute machine due to the limitation of the memory.

REFERENCES

[1] http://fimi.ua.ac.be/data/ "chess.gz".

[2] Jiawei Han & Micheline Kamber, "Data Mining: Concepts and Techniques," Morgan Kaufmann Publishers, An Imprint of Elsevier, Org. ISBN: 1-55860-489-8 Indian Reprint ISBN: 81-8147-049-4,PP 452-481,(2004) .

[3] Ashwin Garg ,Ashish Mangala, Neelam Gupta, "PBIRCH:A Scalable Parallel Clustering Algorithm for Incremental data"**,** 10th International Database Engineering and Applications Symposium (IDEAS'06), 0-7695-2577-6/06 IEEE, (2006).

[4] Kittisak Kerdprasop and Nittaya Kerdprasop, "A lightweight method to parallel k-means clustering," International Journal of Mathematics and Computers in Simulation, Issue 4, Volume 4, (2010).

[5] S.V. Adve, "Parallel Computing Research at Illinois: The UPCRC Agenda," 2008 .

[6] Barney & Blaise "Introduction to Parallel Computing," Lawrence Livermore National Laboratory, http://www.llnl.gov/computing/tutorials/parallel_comp/. 2007.

[7] Cluster Computing www.ccgrid.org/types-of-computer-clusters.html.

[8] http://www.mcs.anl.gov/research/projects/mpich2.

[9] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker,Jack Dongarra, "MPI: The Complete Reference," The MIT Press,Cambridge, Massachusetts London, England, PP 9-14, (1996).

[10] S.N. Tirumala Rao, E.V. Prasad and N.B. Venkateswarlu, "A Scalable k-means Clustering Algorithm on Multi-Core Architecture," International Conference on Methods and

Models in Computer Science Proceeding , PP 1-9, 14-15, (Dec.' 2009).

[11] S.N. Tirumala Rao, E.V. Prasad, N.B. Venkateswarlu, "A Critical Performance Study of Memory Mapping on Multi-Core Processors: An Experiment with k-means Algorithm with Large Data Mining Data Sets," International Journal of Computer Applications (0975 – 8887) Volume 1 – No. 9, (2010).

[12] Tian Zhang, Raghu Ramakrishnan, Miron Livny, "BIRCH: A New Data Clustering Algorithm and its Applications," Volume 1, Issue 2, pp 141-182, (1997).

[13] R.O. Duda, P.E. Hart, "Pattern Classification and Scene Analysis," Wiley, (1973).

[14] W. Gropp, E. Lusk, A. Skjellum, "Using MPI: Portable Parallel Programming with the Message Passing Interface," The MIT Press, Cambridge, MA (1996).

[15] M. Snir, S.W. Otto., S. Huss-Lederman, D.W. Walker, J. Dongarra., "MPI: The Complete Reference," The MIT Press, Cambridge, MA (1997).

[16] Peter S.Pacheco, "A User_s Guide to MPI", (2008).

[17] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, Jack Dongarra, "MPI: The Complete Reference," The MIT Press, Cambridge, Massachusetts London, England, (2002).

[18] Kilian Stoel and Abdelkader Belkoniene, **"**Parallel *k*/*h*-Means Clustering for Large Data Sets," Springer-Verlag Berlin Heidelberg, Euro-Par'99, LNCS 1685. pp. 1451-1454, (1999).

.

[19] Wenbin Fang, Ka Keung Lau, Mian Lu, Xiangye Xiao, Chi Kit Lam, Philip Yang Yang, Bingsheng He, Qiong Luo, Pedro V. Sander, and Ke Yang, **"**Parallel Data Mining on Graphics Processors," Technical Report HKUSTCS0807, (Oct 2008).

[20] Kittisak Kerdprasop and Nittaya Kerdprasop, "A lightweight method to parallel k-Means clustering," International Journal of Mathematics and Computers in Simulation, Issue 4, Volume 4, pp 144-153, (2010).

[21] A. Raghuvira Pratap, J. Rama Devi , K. Suvarna Vani and K Nageswara Rao, "An Efficient Density based Improved K-Medoids Clustering algorithm," International Journal of Advanced Computer Science and Applications(IJACSA), Vol. 2, No. 6, ,pp 49-54, (2011).

[22] David Pettinger and Giuseppe Di Fatta "Space Partitioning for Scalable K-Means," Ninth International Conference on Machine Learning and Applications, pp 319-24, IEEE (2010).

[23] S.Hari Ganesh and C.Chandrasekar, "A Parallel Computing Data Mining and Enhanced K-means Algorithm for Detecting Protein Sequence," International Journal of Computing Technology and Information Security Vol.1, No.1, pp.56-61, (March, 2011)

[24] S.N. Tirumala Rao, E.V. Prasad and N.B. Venkateswarlu, "A Scalable k-means Clustering Algorithm on Multi-Core Architecture," Proceeding: International Conference on Methods and Models in Computer Science, pp 1 – 9, (2009).