OPEN ACCESS
ISSN 2356-5462
socj.telkomuniversity.ac.id/ijoict

Intl. Journal on ICT
Vol. 2, Issue. 2, December 2016. pp. 27-36
doi:10.21108/ijoict.2016.22.86

Comparative Study between Parallel K-Means and Parallel K-Medoids using Message Passing Interface (MPI)

Agnis Nanda[1], Fhira Nhita[2], Fitriyani[3] , Deni Saepudin[4]

1,2,3,4 *Telkom University*
Bandung, Indonesia

[1] agnisnandars@gmail.com
[2] fhiranhita@telkomuniversity.ac.id
[3] fitriyani@telkomuniversity.ac.id
[4] denisaepudin@telkomuniversity.ac.id

Abstract

Data mining is a combination technology for analyze a useful information from dataset using some technique such as classification, clustering, and etc. Clustering is one of the most used data mining technique these day. K-Means and K-Medoids are one of clustering algorithms that mostly used because it's easy implementation, efficient, and also present good results. Besides mining information, the needs of time spent when mining data is also a concern in today era considering the real world applications produce huge volume of data. This research analyzed the results from K-Means and K-Medoids algorithm and time performance using High Performance Computing (HPC) Cluster to parallelized K-Means and K-Medoids algorithms and using Message Passing Interface (MPI) library. The results shown that K-Means algorithm gives smaller Sum Squared of Error (SSE) than K-Medoids. And also parallel algorithm that used MPI gives faster computation time than sequential algorithm.

Keywords: Clustering, K-Means, K-Medoids, Sum Squared of Error, Message Passing Interface

I. Introduction

Nowadays, data generation advancement are massively and rapidly developed. Collecting any data is possible everywhere and anywhere. There are a lot type of data with various amount of data which stored on data warehouse like sales production, satellite orbit lane, disease data, and various data type from many disciplines.

Gathering information and processed into knowledge could be done with data mining technique. Data mining is a technology that combined data analysis method with several massive-data processing algorithm. Data mining were also used to help find and analyze new information from data that ever used with different method. Clustering is one of data mining technique.

Clustering is a data mining technique which very useful for real problems [9]. The concept of clustering is similarity based on distance on a same group and difference on another group [9,10]. There were a lot of clustering algorithms that could be used. Selection of clustering algorithm could be based on fata type or use of data.

Today's data generation also have a dimension that could get up to thousands of dimensions. The problems is how we could process thousands dimensions data not only with a great accuracy but also with a shortest time possible. High Performance Computing was considered capable of supporting data mining process. Use of HPC in data mining were widely used. Just like a research done by Jing Zhang, Gongqing Wu, Xuegang Hu, Shiying Li, Shuilang Hao titled "A Parallel K-means Clustering Algorithm with MPI"[1]. Other

Received on Aug 2016. Accepted on Nov 2016

researches had already compare between K-Means and K-Medoids [8,9]. We propose HPC Cluster approach to implement K-Means and K-Medoids in parallel platform. Parallel data clustering using Message Passing Interface (MPI) were done in this research to get a high accuracy and low computational time for clustering result on data mining process.

II. RELATED WORK

A. K-MEANS CLUSTERING

K-Means algorithm is the mostly used clustering algorithm [1]. First, determine the amount of K cluster. Pick the initial centroid cluster randomly and K-Means algorithm will repeat this steps until the centroid don't change[2,9]:

TABLE
K-MEANS ALGORITHM

K-Means Algorithm
Input: Data, K Cluster
Output: K Centroid
1: Choose K point as initial centroid.
2: **repeat** 3: From the defined centroid, assigned each objects to closest centroid by using Euclidean Distance formula.
4: Compute new clusters centers
5: **until** centroid don't change

B. K-MEDOIDS CLUSTERING

Partitioning Around Medoids (PAM) or known as K-Medoids has similar algorithm with K-Means Clustering [3,9]. The difference is, centroid that used in K-Means is the means of closest centroid in C cluster but in K-Medoids, the centroid itself with minimum cost:

TABLE II
K-MEDOID ALGORITHM

K-Medoids Algorithm
Input: Data, K Cluster
Output: K Centroid
1:Choose K point as initial centroid.
2: **repeat** 3: From the defined centroid, assigned each objects to closest centroid by using Euclidean Distance formula.
4: Choose K point non medoid randomly
5: Compute total cost, if smaller then change centroid
6: **until** centroid don't change

C. MESSAGE PASSING INTERFACE (MPI)

MPI is standard library by using message-passing mechanism for parallelized the algorithm to support parallel computing[1]. Parallel programing using MPI is defined clearly by choosing what functions to used. This is the general structured of MPI programming:

1) MPI Include file
2) Start serial code
3) Initialize MPI
4) Do work & make message passing calls
5) Terminate MPI environtment
6) Continue the serial code
7) End program.

D. PARALLEL K-MEANS AND K-MEDOID CLUSTERING

Parallelized in basic have same meaning in partition data so data can be execute in same time[4]. K-Means and K-Medoids Clustering have same intensive computation, it is in compute the distance. In parallel system, the main idea is partition to each processes so the processes will have same amount of data and can do the processes in same time.

Each computer can do the algorithm, and have centroid in each process. After that, the result of each processe will be merged in head node. Parallel K-Means and K-Medoids algorithm will be explain in Table 3 and 4.

TABLE III
PARALEL K-MEANS ALGORITHM

Parallel K-Means Algorithm
Input: Data, K Cluster
Output: K Centroid
1: MPI_INIT// start MPI Procedure
2: Read N object from file
*/start parallel proccess by divide same amount of object to each processes/**
3: **repeat**
4:Choose K point as intial centroid randomly
5: Initiate each object to the closest centroid by using Euclidean Distance Formula
6: **until** centroid don't change
*/merge centroid procedure /**
7: Generate cluster id to each object
8: Generate new centroid cluster by centroid result in each processes
9: Generate final centroid
10: MPI_Finalize() // Terminate MPI Process

TABLE IV
PARALEL K-MEDOID ALGORITHM

Paralel K-Medoids Algorithm
Input: Data, K Cluster
Output: K Centroid
1: MPI_INIT// start MPI procedure
2: read N object from file
*/start parallel proccess by divide same amount of object to each processes/**
3: **repeat**
4:Choose K point as intial centroid randomly
5: Initiate each object to the closest centroid by using Euclidean Distance Formula
6: Choose K point non medoid randomly
7: Compute total cost K Medoid and K non Medoid then .
8: **until** centroid don't change
/merge the result from each processes/
9: Generate cluster id for each processes
10: Generate centroid new cluster
11: Generate final centroid
12: MPI_Finalize() // terminated MPI process

E. Cluster Evaluation

Cluster evaluation is a part of clustering analysis. Because the algoritm using Euclidean Distance formula as closeness measurement, so the objective function that can be used for measure the quality of cluster is Sum Squared of Error (SSE). the explanation of the formula is explained below [2]:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2 \quad (1)$$

F. Parallel Performance Evaluation

Parallel performance improvement, can be measure using speedup, performance improvement and speedup [5]. This three evaluation will measure how good the number of processor by the computation time. *Speedup* measure how fast the time that parallel algorithm used than sequential algorithm. Speedup formula can be wrote as:

$$speedup = \frac{sequentialtime(s)}{paralletime(s)} \quad (2)$$

Performance improvement, describe the relation of improvement parallel process than sequential process. Performance improvement formula can be wrote as:

$$performanceimprovement = \frac{sequentialtime(s) - paralleltime(s)}{sequentialtime(s)} \quad (3)$$

Efficiency estimated how processor be used for processing comparing the amount of work that used for communicate and synchronized. Efficiency formula can be wrote as:

$$eficiency = \frac{sequentialtime(s)}{no.processor \times paralleltime(s)} \quad (4)$$

G. DATASET

The dataset that used in this research was selected from UCI Machine Learning Repository [6] and KentRidge Biomedical Dataset [7]. We expected, this various type of dataset will give more information whether the number of attribute or the number the record will affect the time of processing.

TABLE V
DESCRIPTION OF DATASET

No	Data Sets	Number of Attribute	Number of Record	Information	Size
1	Prostate Cancer	12600	102	Normal, Tumor	14,038 KB
2	Skin	4	245057	Skin, Non Skin	10,079 KB
3	Ovarian Cancer	15154	253	Normal, Cancer	32,943 KB
4	Tumor	2000	62	Tumor, Normal	1,402 KB
5	Letter	6	20000	A-Z	3,052 KB
6	Segment	13	2311	7 Segment	440 KB

H. EXPERIMENTAL ENVIRONMENT

The hardware platform in this paper use HPC Cluster with total 12 Compute node and 48 core processors intel i7 @3.0 GHz, 96 GB memory, and the network environment use UTP Cable LAN.

I. PERFORMANCE EVALUATION

In this research will perform the evaluation of both K-Means and K-Medoids algorithm such as Sum Squared of Error (SSE), sequential and parallel computation time and also the performance of using MPI to parallelize the algorithm.

III. RESEARCH METHOD

The proposed method in this research will compare the algorithmn both sequential and parallel process. The process in this section will divide In five steps as describe in Fig. 1, there are (1) pre-processing data, (2) training the dataset by using K-Means and K-Medoids algorithm, (3) Parallelized the algorithm with MPI, (4) Validation, (5) Performance Analysis.

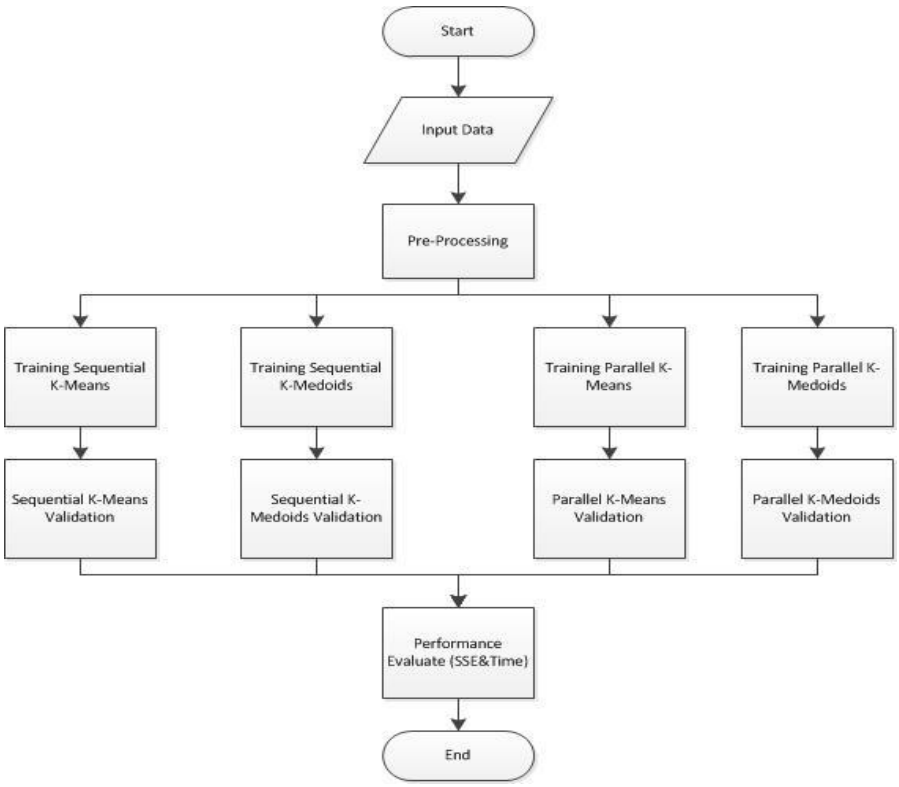

Fig.1 Design of System

The explanation of fig 1 explained below :

A. Pre-processing Data

Pre-processing is a process to prepare the dataset from raw data until the data ready to be processed. The proposed of processing is to minimize the possibility of error. In this research the selected data do not have missing value so the preprocessing is only normalized the dataset. The dataset will normalized with the formula below [11]:

$$v' = \frac{v - \min_s}{(\max_s - \min_s)} \tag{5}$$

Where :

v' = normalized data

v = raw data

max_s= maximum data in dimension-s

min_s= minimum data in dimension-s

B. Training the dataset by using both K-Means and K-Medoids Algorithm and Parallel method

In this step will build cluster from data. The cluster will have membership and centroid. The centroid will be used in validation step. We did training in every method and every algorithm.

C. Parallelized algorithm with MPI

This step has same mecanishm with sequential step but the difference is the algoritmn will be parallized with MPI so the computation time will be reduced.

D. Validation

This step will be inputted by different data from before. It will give membership result by using the centroid from training step. We did training in every method and every algorithm.

E. Performance Evaluation

In this steps will evaluate the result of clustering, computation time and also the use of processor in parallelize algorithm. The number processor that used for this research is 2, 4,6 and 8.

IV. Results And Discussion

A. Clustering Performance

As have been mentioned before, the cluster result from both algorithm will be evaluated by using SSE. Table VI shown the results of SSE from K-Means and K-Medoids algorithm in each dataset. As we can see in Table VI the SSE from K-Means is smaller than K-Medoids algorithm. So the centroid from means is closer than random object that being centroid.

TABLE VI
SSE RESULTS

Dataset	K-Means	K-Medoids
Tumor	4477.971	7471.021
Prostate	31381.896	39289.398
Ovarian	98469.82	195812.703
Skin	23522.475	66004.984
Letter	3857.646	4796.093
Segment	693.871	1043.502

B. Time Evaluation of Sequential Computation

Clustering performance can also be seen in sequential computation time. Fig 2 shown that K-Means computation time is faster than K-Medoids. This is matching with the complexity that K-Means and K-Medoids have[8]. The comparison of computation time shown in Fig 2.

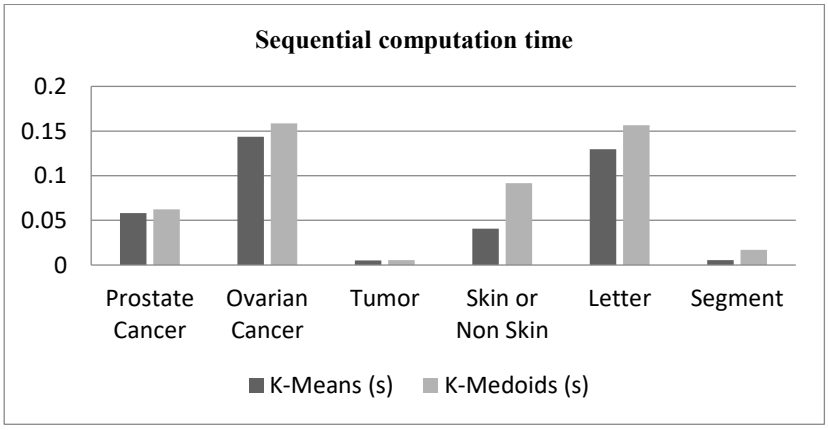

Fig.2 Comparison of Sequential Time

C. Parallel Performance Evaluation

This section will evaluate the speedup, performance improvement and efficiency from the use of processor. Fig 3,4 and 5 shows the speedup, performance improvement and efficiency result in both algorithm. The result below was from the formula (2,3,4).

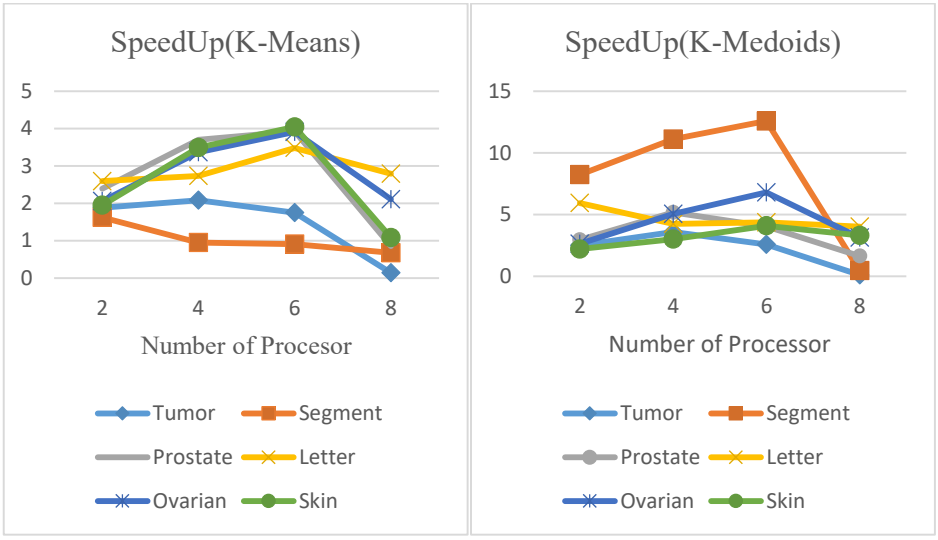

Fig.3 Speed Up Graphic

In Fig 3 shows that every dataset has different 'best' speedup but all have same 'worst' speedup. In every eight processor, the speedup always down. It is shown that the dataset are not big enough to parallelized so the algorithm take more time to communicate and synchronized. For performance improvement will shown in fig 4.

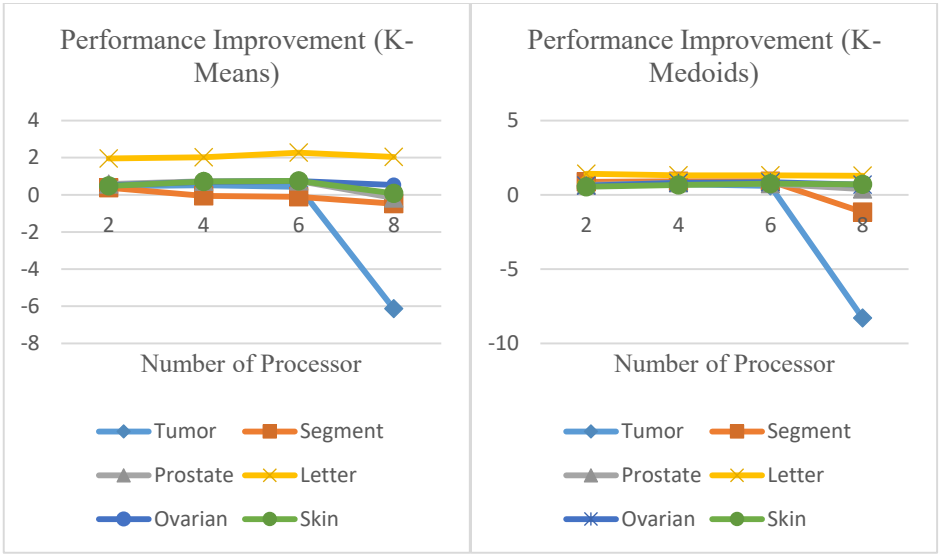

Fig.4 Performance Improvement Graphic

In fig 4 shows that every processor have no much different in improvement. But in contrast, the performance for Tumor dataset, both K-Means and K-Medoids algorithm are not improve.

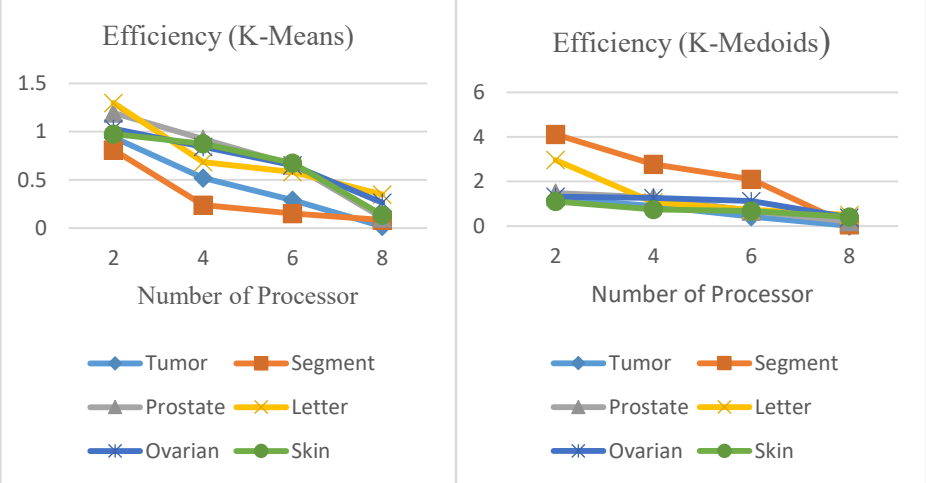

Fig.5 Efficiency Graphic

In Fig 5, the graphic clearly shows that the efficiency by using two processor is the most efficient for that range size of data.

V. Conclusion

The experiment results shows that K-Means algorithms has better performance than K-Medoids algorithm. But in reverse, K-Medoids has better algorithm to be parallelized than K-Means because K-Medoids algorithm has more computing section than K-Means algorithm. For paralleling the data, MPI gives fastest time for computing but it depends on how big the data and the core we used. For our study, the data not big enough to used 8 cores but 2 cores gives better performance than using serial algorithm.

ACKNOWLEDGMENT

This work was supported in part by Telkom University research grant.

REFERENCES

1. Jing,Zhang., Gongqing, Wu., Xuegang, Hu., Shiying, Li., Shuilong, Hao. (2011) A Parallel K-Means Clustering Algorithm with MPI. International Symposium on Parallel Architectures, Algorithms and Programming, 2011 IEEE
2. Tan, Pang-Ning., Steinbach,Michael., Kumar,Vipin.(2006) Introduction to Data Mining.
3. Jiawei, Han., Kamber, Micheline.(2001) Data Mining Concepts and Technique.

F. Lusk, N. Doss, A. Skjellum. (1996) A High-Performance, Portable Implementation of the MPI Message Passing Interface. Parallel Computing. vol.22, pp 789-828.

4. Ahmad Firdaus Ahmad Fadzil, Noor Elaiza Abdul Khalid, Mazani Manaf. (2011) Scaling Perormance of Task-Intensive Applications via Mapreduce Parallel Processing. Faculty of Computer anda Mathematical Science, UiTM Shah Alam, Selangor, Malaysia.

5. C. Blake, E. Ceogh, C. Merz. (1996) UCI Repository of Machine learning databases. Irvine: Departement of Information and Computer Science, University of California.
6. KentRidge Biomedical Dataset Repository.Retrieved 13 August 2014, from http://datam.i2r.a-star.edu.sg/datasets/krbd/
7. S Singh, Shalini,. N. C, Chauhan.: K-means v/s K-medoids.(1996) A Comparative Study. National Conference on Recent Trends in Engineering & Technology.
8. T. Soni Madhulatha: Comparison Between K-Means and K-Medoids Clustering. (2011) International Journal of Advanced Computing (IJAC) Vol 3
9. Hesam T. Dasthi., Tiago Simas., Rita A. Ribein., Amir Assadi,. And Andre Moitinho. (2010) MK-Means – Modified K-Means clustering algorithm.WCCI 2010 IEEE World Congress on Computational Intelegence. CCIB Barcelona, Spain.
10. Vilasaki, N. Karthikeyani., K. Thangavel. (2009) Impact of Normalization in Distributed K-Means Clustering. International Journal of Soft Computing.