Parallel Customer Clustering: A Computational Performance Study

Review of Integrative Business & Economics Research

Nittaya Kerdprasop*
Naritsara Tharaputh
Supagrid Tangsermsit
Chonlawit Sepasiraporn
Kittisak Kerdprasop

Data Engineering Research Unit, School of Computer Engineering,
Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand

ABSTRACT

In the era of big data, automatic formation of customer data into meaningful groups is a very computational time-consuming task. Parallelization the data computing process by distributing subsets of data to different processing nodes in a concurrent manner is one efficient solution for the big data computation. In this research work, we present the empirical study results of customer data clustering using partitioning method based on the k-means algorithm for numeric data and the k-modes algorithm for categorical data. Implementation of parallel and sequential k-means and k-modes has been done through the Erlang programming language. The source codes are also openly available in the appendix of this paper. Running results of parallel k-means and k-modes have been compared against the sequential computation. The results show good performance of k-means at a certain point of parallelization with small to moderate amount of data, but the computational performance decreases after data becoming big. The parallel k-modes, on the contrary, show good performance on a wide range of data size from the small to a big one. From our computational performance study results, the parallel scheme has been proven a possibly solution to improve customer clustering to support predictive marketing over huge business databases.

Keywords: Parallel computation, Concurrent programming, Data clustering, k-Modes.

1. INTRODUCTION

Analytics and intelligence are emerging key technologies for modern businesses. Predictive marketers use analytic tools to gain the ability to understand customers' behaviors that helps planning marketing strategy specific to a unique group of customers. The intelligent analytical tools are basically developed from the research advancement in the inference statistics, machine learning, and data mining fields (Hair, 2007; Holsapple et al., 2014). These tools had been applied in many business sectors including retail industry (Chen et al., 2012), telecommunication (Arumawadu et al., 2015), multimedia on demand (Hung and Tsai, 2008), banking (Hsieh, 2004), and healthcare (Chen et al., 2012).

Market segmentation by grouping customers showing some similar behaviors is one analytical task extensively performed by marketing experts. Customer grouping has

ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

normally been done by applying one of the clustering techniques (Brusco et al., 2003; Chang and Tsai, 2011; Ho et al., 2012; Singh et al., 2014; Yao et al., 2010). Among many available clustering techniques, k-means is the most applicable one (Halkidi et al., 2001). The popularity is due to its simplicity and effectiveness for small to medium data sets, but its efficiency in terms of computational time can degrade when data become big.

In this paper, we thus experimental study the characteristic of k-means when it has been modified to run in a parallel scheme. The advantages of parallelization are to reduce computational time and to handle big data (Wang et al., 2004; Kerdprasop and Kerdprasop, 2010; Ren et al., 2013; Tao et al., 2015). In case of numerical data, we do parallel experiment with the k-means algorithm (MacQueen, 1967). For the categorical data, the k-modes algorithm (Huang, 1998), which is an extension of k-means, has been used in our experimentation. Conventional k-means and k-modes are reviewed in the next section. The parallel versions are presented in Section 3. The experimental results are shown in Section 4. The conclusion is in the last section of this paper. The source codes of k-modes implementation, in both sequential and parallel forms, using the Erlang programming language are also available in the Appendix.

2. SEQUENTIAL CLUSTERING

Clustering is the iterative process of finding groups of data subsets in such a way that data in the same group are similar to each other than those in the other groups. By means of clustering, some customer behaviors can be revealed. For instance, forming clusters from the retail database can show customer segments sharing the same buying patterns. A simple and effective way of clustering is partitioning data into disjoint subsets using the k-means algorithm (MacQueen, 1967). This algorithm works well with numeric data because data similarity measurement is based on the Euclidean distance computation. For categorical data, k-modes algorithm (Huang, 1998) is more suitable than k-means because the match/mismatch in attribute data value is consider instead of the distance computation and the mode of each cluster is reported rather than the cluster mean value (He et al., 2011). The algorithms of conventional k-means and k-modes are given in Figure 1.

Input: Customer Data (numeric values) **Output:** Clusters of customer subsets **Steps:** 1. Given a desired number of clusters, *k*, randomly select *k* data records to be initial cluster centers. 2. Repeat until all centers do not change. 2.1 Assign each data record to the closest cluster by considering Euclidean distance of the record to every cluster center. 2.2 Compute a new cluster center	**Input:** Customer Data (categorical values) **Output:** Clusters of customer subsets **Steps:** 1. Given a desired number of clusters, k, randomly select k data records to be initial cluster center. 2. Repeat until all centers do not change. 2.1 Assign each data record to the closest cluster by considering similarity of data fields (similar value = 0; dissimilar = 1). 2.2 Compute a new cluster center using a mean value of the cluster members.

ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

using mean value of the cluster members. 3. On completion, report the mean value and members of each cluster	3. On completion, report the mode value and members of each cluster

Figure 1. Clustering algorithms: k-means (left) and k-modes (right).

3. PARALLEL COMPUTATIONAL PERFORMANCE STUDY

To improve the performance of customer clustering, we propose to use the scheme of parallel computation. On parallel computing, large data set is to be split into several subsets and these data subsets can then be computed for finding closest cluster concurrently. The algorithms for parallel k-means and parallel k-modes are given in Figure 2. Our implementation of parallel k-modes using Erlang language is also shown in Appendix.

Input: Customer Data (numeric values) **Output:** Clusters of customer subsets **Steps:** 1. Given a desired number of clusters, *k*, randomly select *k* data records to be initial cluster centers, C. 2. Divide data equally into P subsets. 3. Sending C and P to each processor node to compute distance and assign cluster for each data record. 4. Receive cluster members back from all processor nodes. 5. Compute new center for each cluster. 6. If the new center is different from the previous center, then go back to step 2; otherwise, stop the process. 7. Report the mean value and members of each cluster	**Input:** Customer Data (categorical values) **Output:** Clusters of customer subsets **Steps:** 1. Given a desired number of clusters, k, randomly select k data records to be initial cluster center, M. 2. Divide data equally into P subsets. 3. For each data subset, D, create a new process, and send D and M to compute similarity and form a cluster. 4. Receive cluster members back from all processor nodes. 5. Compute new mode as a cluster center 6. If the new center is different from the previous center, then go back to step 2; otherwise, stop the process. 7. Report the mode value and members of each cluster

Figure 2. Parallel clustering algorithms: k-means (left) and k-modes (right).

4. EXPERIMENTAL RESULTS

To study characteristics of parallel clustering over numerical values using k-means algorithm and over categorical values using k-modes algorithm, we generate different sizes of synthetic data. For parallel k-means, data are two dimensional integers randomly generated. Examples of data are: [1675,874], [2249,3886], [1124,2603], [4589,3606], [4236,4792], [4829,170]. The number of data instances has been increased from 10, 50, 100, 200, 300, 400, 500, 1000, 2000, 3000, 4000, 5000 up to 10000 instances. For parallel k-modes, data are five dimensions and all dimensions are categorical values, for instance, [sunny,hot,high,false,no], [sunny,hot,high,true,no], [cloudy,hot,high,false,yes], [rainy,mild,high,false,yes], [rainy,cool,normal,false,yes].

ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

Clustering experimentation over both numerical and categorical data sets has been done on personal computer with 4 processing cores, 8 GB RAM, and 64 bit Windows operating system. We set k-means and k-modes running to group data into 5 clusters. Characteristics of parallel k-means when dealing with increasing data are demonstrated in Table 1 and Figure 3. Parallel k-modes performance, as compared to the sequential running, is illustrated in Table 2. The comparison of parallel versus sequential is also graphically shown in Figure 4.

Table 1. Running time of parallel k-means comparative to the sequential computation.

Number of Data Instances	Running Time (seconds)		Time Difference (Ts-Tp)	Speedup (%)
	Sequential k-means (Ts)	Parallel k-means (Tp)		
10	0.004	0.001	0.003	75.0000
50	0.007	0.004	0.003	42.8571
100	0.009	0.005	0.004	44.4444
200	0.022	0.006	0.016	72.7273
300	0.025	0.011	0.014	56.0000
400	0.027	0.013	0.014	51.8519
500	0.03	0.015	0.015	50.0000
1,000	0.039	0.021	0.018	46.1538
2,000	0.061	0.034	0.027	44.2623
3,000	0.086	0.053	0.033	38.3721
4,000	0.108	0.07	0.038	35.1852
5,000	0.135	0.085	0.05	37.0370
10,000	0.264	0.179	0.085	32.1970

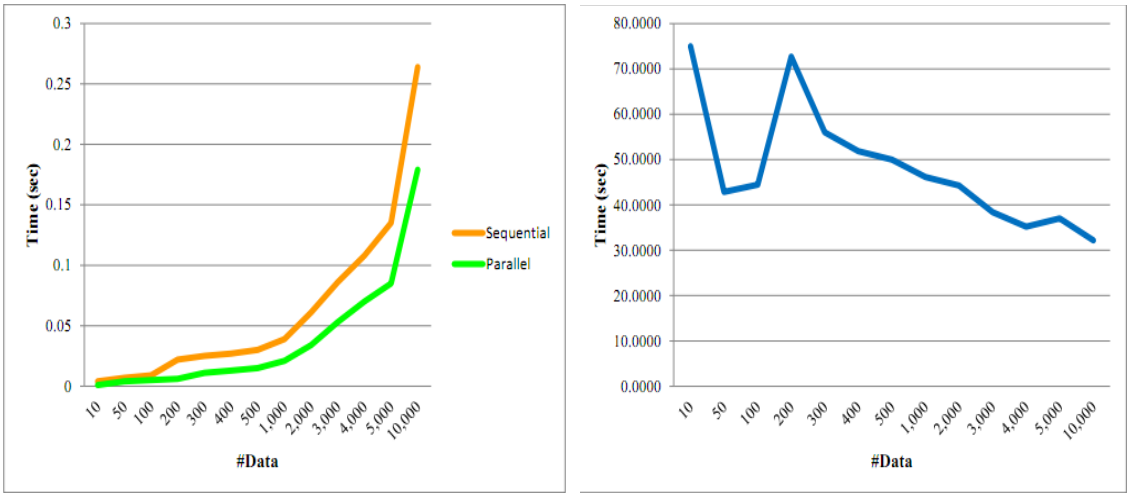

Figure 3. Running time comparison of parallel versus sequential k-means clustering (left) and the advantage of parallel computation in terms of time reduction when the number of data increases (right).

ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

It can be seen from the Table 1 and Figure 3 (left) that parallel k-means runs faster than the sequential version. But at small data sizes (50 and 100 data instances), the time reduction of parallel k-means, computed as speedup percentage over the sequential k-means, drops from over 70 to around 40 percent. When data size is more than 1000 instances, the speedup of parallel k-means drops below 40 percent. This may due to the distance computation of k-means that has to be computed for each data instance to every cluster center. That means the more clusters to form, the more time consuming of parallel k-means.

Table 2. Running time of parallel k-modes comparative to the sequential computation.

Number of Data Instances	Running Time (seconds)		Time Difference (Ts-Tp)	Speedup (%)
	Sequential k-modes (Ts)	Parallel k-modes (Tp)		
10	0.0372	0.0240	0.0132	35.4839
50	0.4668	0.2260	0.2408	51.5853
100	1.6090	0.7470	0.8620	53.5736
200	6.3100	2.7600	3.5500	56.2599
300	14.2780	5.9900	8.2880	58.0473
400	25.1760	10.8600	14.3160	56.8637
500	39.7888	16.7062	23.0826	58.0128
1,000	157.6340	65.9007	91.7333	58.1939
2,000	669.7640	261.8640	407.9000	60.9020
3,000	1441.6200	593.5240	848.0960	58.8294
4,000	2620.0820	1050.8130	1569.2690	59.8939
5,000	3979.5150	1666.7050	2312.8100	58.1179
10,000	16111.4580	6796.5500	9314.9080	57.8154

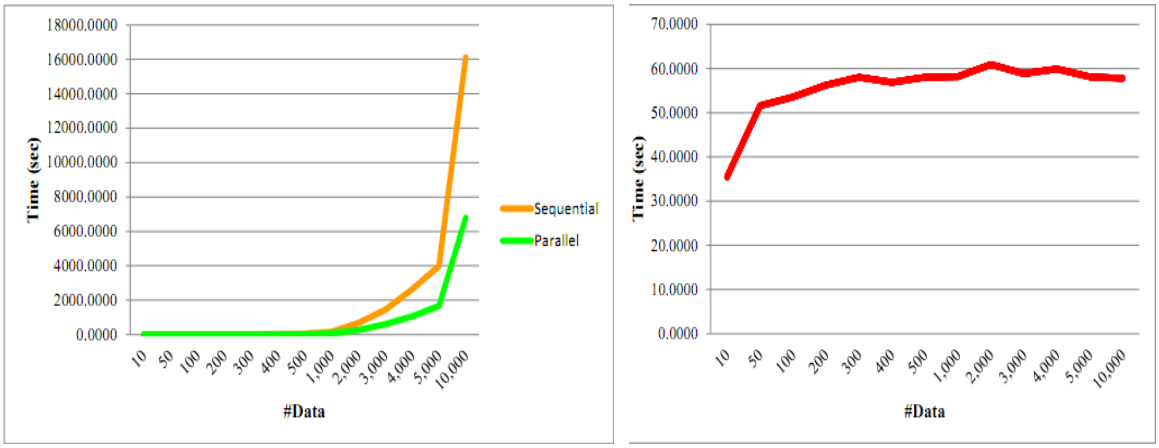

Figure 4. Running time of parallel versus sequential k-modes clustering (left) and the advantage of parallel computation in terms of time reduction when the number of data increases (right).

ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

Parallel k-modes clustering shows a different characteristic than the parallel k-means in such a way that the speedup in terms of running time reduction is quite stable at 60 percent (Figure 4, right). This experimental results reveal the advantage of parallel computation of k-modes clustering when dealing with categorical data. It may be inferred from this study that clustering over customer data that are all categorical values should be performed with the parallel k-modes.

5. CONCLUSION

Clustering is commonly used for customer segmentation in many business industries. For large organization that has to deal with big data computation problem, we suggest parallel computation as an efficient and effective solution. However, clustering can be performed through a wide range of algorithmic selection. We propose in this work the simple partitioning-based method through the application of k-means and k-modes algorithms. The k-means is good at computing numerical values, whereas the k-modes is an extension of k-means to handle categorical case. We thus performed the computational experiments over k-means and k-modes to observe their characteristics when they were modified to run in parallel processors. We found from the experimental results that parallel k-means is sensitive to the amount of data. The more data to compute, the less speedup of the algorithm. This situation is not true for the parallel k-modes. When running in parallel scheme, k-modes is less sensitive to the increasing amount of data. From this finding, we are therefore planning to further our study of parallel k-modes clustering over real large customer data.

APPENDIX

The following source codes of sequential and parallel k-modes are in the Erlang programming language format.

Sequential k-modes

```
-module(kmodes).
-compile(export_all).
attr() -> [[sunny,overcast,rainy] , [hot,mild,cool], [high,normal],[false,true],[no,yes] ] .
data() -> {_, Data} = file:consult("data.dat"), file:close("data.dat"),Data.
leng_att() -> length(attr()).
%----------------Start Program----------------
time(K) ->{Time , _Res} = timer:tc(kmodes , kmo , [K]),
            io:format("~n~n---- Time Start ----~n~n"), io:format("Seq Group= ~w ~n" , [K]),
            io:format("Time = ~w Second~n" , [Time/1000000]),
            io:format("Data Length = ~w Data~n" , [length(data())]),
            io:format("~n~n---- Time End ----~n~n").
kmo(K) -> Data=data(),First_Mode=[lists:nth(Num_of_Mode,Data)||Num_of_Mode<-lists:seq(1,K)],
          programe_start(K,Data,First_Mode,[]).

programe_start(K,Data,Mode,Save_data) ->
      Data_of_Count=[count_attribute_of_All_Data(Data,Mode_Select)||Mode_Select <- Mode],
      Data_of_Select=[find_group(select_of_attribute(Seq,Data_of_Count),Seq)||Seq <- lists:
                        seq(1,length(Data))],
      New_Mode=[partition_of_data(Data_of_Select,K_of_Group)||K_of_Group<-lists:seq(1,K)],
      if   Save_data==Data_of_Select -> show_data(sortTuple(Data_of_Select));
           Save_data=/=Data_of_Select -> programe_start(K,Data,New_Mode,Data_of_Select)
      end
```


ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

```
count_attribute_of_All_Data(Data,Mode)->[lists:sum(count_of_Each_Data(Data_Select,Mode))||Data_
        Select<-Data].
count_of_Each_Data([],[])->[];
count_of_Each_Data([DH|DT],[MH|MT])->
      if   DH==MH->[0|count_of_Each_Data(DT,MT)];
           DH=/=MH->[1|count_of_Each_Data(DT,MT)]
      end.
%-------------Select Group and Find Group -------------
select_of_attribute(Seq,Data)-> [lists:nth(Seq,Data_Select)||Data_Select<-Data].
find_group(Data,Seq) -> Min_of_Lists=lists:min(Data),find_group_data(Data,Min_of_Lists,1,Seq).
find_group_data([],_,Count,_) -> Count;
find_group_data([Data_Head|Data_Tail],Min,Count,Seq)->
      if   Data_Head==Min -> find_group_data([],Min,{Count,lists:nth(Seq,data())},Seq);
           Data_Head=/=Min -> find_group_data(Data_Tail,Min,Count+1,Seq)
      end.
%-------------- Partition Data--------------
partition_of_data(Data,K)-> {H,_}=lists:partition(fun(X)-> element(1,X) =:= K    end,
        Data),newmode(list_group(H)).
list_group(H)->[element(2,D)||D<-H].

%-----------------New Modes ------------------
newmode(Data)->[partition([lists:nth(Seq,D)||D<-Data],Seq)||Seq<-lists:seq(1,leng_att())].
partition(Data,Seq)-> Parti=[partition_of_mode(Data,Att)||Att<-lists:nth(Seq,attr())],
        map_of_mode_and_att(Parti,lists:nth(Seq,attr())).
partition_of_mode(Data,Att)-> {H,_}=lists:partition(fun(X)-> X =:= Att    end, Data),length(H).
map_of_mode_and_att(Data,Att) -> map_data(lists:max(Data),Data,1,Att).
map_data(_,[],_,Data)-> Data;
map_data(Max,[Data_Head|Data_Tail],Count,Att) ->
      if   Data_Head==Max  -> map_data(Max,[],Count,lists:nth(Count,Att));
           Data_Head=/=Max        -> map_data(Max,Data_Tail,Count+1,Att)
      end.
%---------------end New Modes --------------
sortTuple(P) -> lists:sort(fun (T1,T2) -> element(1,T1) < element(1,T2)    end, P).
show_data([D|T]) -> io:format("~w~n",[D]),show_data(T);
show_data([]) -> io:format("---END--~n").
%---------------End Sequential k-Modes --------------
```

Parallel k-modes

```
-module(pkmodes).
-compile(export_all).
attr()->[[sunny,overcast,rainy] , [hot,mild,cool], [high,normal],[false,true],[no,yes] ] .
data()->{_, Data} = file:consult("data.dat"),
file:close("data.dat"),Data.
kmo(K) -> Data=data(), First_Mode=[lists:nth(Num_of_Mode,Data)||Num_of_Mode<-lists:seq(1,K)],
        NumPar=K,Data_split=mysplit(length(Data) div NumPar,Data, NumPar),    {Time , _Res} =
        timer:tc(pkmodes , programe_start , [K,Data_split,First_Mode]),
        io:format("~n~n---- Time Start ----~n~n"),    io:format("Seq Group= ~w ~n" , [K]),
        io:format("Time = ~w Second~n" , [Time/1000000]),
        io:format("Data Length = ~w Data~n" , [length(data())]),
        io:format("~n~n---- Time End ----~n~n").

programe_start(K,Data,Mode) -> P_ID=sendspawn(K), Group=myloop(P_ID,K,Data,Mode,[]).
myloop(P_ID,K,Data,Mode,Save_data) ->
      Data_of_Select=parallel_start(P_ID,K,Data,Mode,length(P_ID)),
      New_Mode=[partition_of_data(Data_of_Select,K_of_Group)||K_of_Group<-lists:seq(1,K)],
      if Save_data==Data_of_Select -> stop_parallel(P_ID),show_data(sortTuple(Data_of_Select));
         Save_data=/=Data_of_Select -> myloop(P_ID,K,Data,New_Mode,Data_of_Select)
      end
```


ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

```
count_attribute_of_All_Data(Data,Mode) ->
        [[lists:sum(count_attribute_of_each_mode(D,M))||D<-Data]||M<-Mode].
count_attribute_of_each_mode([],[]) -> [];
count_attribute_of_each_mode([DH|DT],[MH|MT]) ->
      if   DH == MH -> [0|count_attribute_of_each_mode(DT,MT)];
           DH =/= MH -> [1|count_attribute_of_each_mode(DT,MT)]
      end.
data_partition(Seq,Data) -> lists:append([lists:nth(Seq,D)||D<-Data]).
mysplit(_,_, 0) -> [];
mysplit(Len, L,Count) -> {H, T} = lists:split(Len, L),
      if   length(T)>=Len ->[ H | mysplit(Len, T, Count-1) ];
           length(T) < Len -> [L|mysplit(Len,T,0)]
      end.
%-------------Select Group and Find Group -------------
select_of_attribute(Seq,Data) -> [lists:nth(Seq,Data_Select)||Data_Select<-Data].

find_group(Data,Seq) -> Min_of_Lists=lists:min(Data),find_group_data(Data,Min_of_Lists,1,Seq).
find_group_data([],_,Count,_) ->Count;
find_group_data([Data_Head|Data_Tail],Min,Count,Seq) ->
      if   Data_Head==Min -> find_group_data([],Min,{Count,lists:nth(Seq,data())},Seq);
           Data_Head=/=Min -> find_group_data(Data_Tail,Min,Count+1,Seq)
      end.
%---------------Partition of data---------------
partition_of_data(Data,K) -> {H,_}=lists:partition(fun(X)-> element(1,X) =:= K    end,
       Data),newmode(list_group(H)).
list_group(H) -> [element(2,D)||D<-H].
%----------------New Modes ----------------
newmode(Data) -> [partition([lists:nth(Seq,D)||D<-Data],Seq)||Seq<-lists:seq(1,length(attr()))].
partition(Data,Seq) -> Parti=[partition_of_mode(Data,Att)||Att<-lists:nth(Seq,attr())],
      map_of_mode_and_att(Parti,lists:nth(Seq,attr())).
partition_of_mode(Data,Att) -> {H,_}=lists:partition(fun(X)-> X =:= Att    end, Data),length(H).
map_of_mode_and_att(Data,Att) -> map_data(lists:max(Data),Data,1,Att).
map_data(_,[],_,Data) -> Data;
map_data(Max,[Data_Head|Data_Tail],Count,Att) ->
      if   Data_Head==Max  -> map_data(Max,[],Count,lists:nth(Count,Att));
           Data_Head=/=Max        -> map_data(Max,Data_Tail,Count+1,Att)
      end.
%----------------end New Modes -------------
sortTuple(P) -> lists:sort(fun (T1,T2) -> element(1,T1) < element(1,T2)    end, P).
show_data([D|T]) -> io:format("~w~n",[D]),show_data(T);
show_data([]) -> io:format("---END---~n").
%-------------------Parallel -------------------
sendspawn(0) -> [] ;
sendspawn(N) -> [spawn(?MODULE, send_recive, [self()]) | sendspawn(N-1) ].
parallel_start(P_ID,K,Data,Mode,Count) -> send_parallel1(P_ID,Data,Mode,Count),
      L=rec_parallel(Count),
      send_parallel2(P_ID,L,lists:seq(1,K),Count),
      S=rec_parallel(Count),
      X=lists:seq(1,length(lists:nth(1,S))),
      XX=mysplit(length(X) div K,X,K),
      send_parallel3(P_ID,S,XX,Count),
      P=rec_parallel(Count),lists:append(P)

find(LH,Data) -> [find_group(select_of_attribute(Y,Data),Y)||Y<-LH].
send_parallel1([PH|PT],[DH|DT],Mode,Count) ->
       PH!{1,DH,Mode,Count},send_parallel1(PT,DT,Mode,Count-1);
      send_parallel1([],_,_,_)-> true.
send_parallel2([PH|PT],Data,[LH|LT],Count) ->
       PH!{2,Data,LH,Count},send_parallel2(PT,Data,LT,Count-1);
```


ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

```
send_parallel2([],_,_,_) -> true.
send_parallel3([PH|PT],Data,[LH|LT],Count) ->
        PH!{3,Data,LH,Count},send_parallel3(PT,Data,LT,Count-1);
send_parallel3([],_,_,_) -> true.

send_recive(PH) ->
receive
stop -> true;
{1,DH,Mode,Count}        ->        L        =        count_attribute_of_All_Data(DH,Mode),PH        !
{Count,L},send_recive(PH);
{2,Data,LH,Count} -> L = data_partition(LH,Data),PH!{Count,L},send_recive(PH);
{3,Data,LH,Count} -> L = find(LH,Data),PH!{Count,L},send_recive(PH)
end.

rec_parallel(0) -> [] ;
rec_parallel(Count) ->
receive
{Count, L} -> [L | rec_parallel(Count-1)]
end.
stop_parallel([PH|PT]) ->      PH ! stop ,  stop_parallel(PT);
stop_parallel([]) -> true.
%----------------End Parallel k-Modes -------------
```

REFERENCES

[1] Arumawadu, H., Rathnayaka, R., and Illangarathne, S. (2015). Mining profitability of telecommunication customers using k-means clustering. *Journal of Data Analysis and Information Processing*, 3(3):63-71.

[2] Brusco, M., Cradit, J., and Tashchian, A. (2003). Multicriterion clusterwise regression for joint segmentation settings: An application to customer value. *Journal of Marketing Research*, XL: 225-234.

[3] Chang, H., and Tsai, H. (2011). Group RFM analysis as a novel framework to discover better customer consumption behavior. *Expert Systems with Applications*, 38(12): 14499-14513.

[4] Chen, Y., Cheng, C., Lai, C., Hsu, C., and Syu, H. (2012). Identifying patients in target customer segments using a two-stage clustering-classification approach: A hospital-based assessment. *Computers in Biology and Medicine*, 42(2): 213-221.

[5] Chen, D., Sain, S., and Guo, K. (2012). Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, 19(3): 197-208.

[6] Hair, J. (2007). Knowledge creation in marketing: The role of predictive analytics. *European Business Review*, 19(4): 303-315.

[7] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3): 107-145.

[8] He, Z., Xu, X., and Deng, S. (2011). Attribute value weighting in k-modes clustering. *Expert Systems with Applications*, 38(12): 15365-15369.

[9] Ho, G., Ip, W., Lee, C., and Mou, W. (2012). Customer grouping for better resources allocation using GA based clustering technique. *Expert Systems with Applications*, 39(2): 1979-1987.

[10] Holsapple, C., Lee-Post, A., and Pakath, R. (2014). A unified foundation for business analytics. *Decision Support Systems*, 64: 130-141.

ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)

[11] Hsieh, N. (2004). An integrated data mining and behavioral scoring model for analyzing bank customers. *Expert Systems with Applications*, 27(4): 623-633.

[12] Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2:283-304.

[13] Hung, C., and Tsai, C. (2008). Market segmentation based on hierarchical self-organizing map for markets of multimedia on demand. *Expert Systems with Applications*, 34(1): 780-787.

[14] Kerdprasop, K., and Kerdprasop, N. (2010). A lightweight method to parallel k-means clustering. *International Journal of Mathematics and Computers in Simulation*, 4(4): 144-153.

[15] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281-197.

[16] Ren, D., Zheng, D., Huang, G., Zhang, S., and Wei, Z. (2013). Parallel set determination and k-means clustering for data mining on telecommunications networks. *Proceedings of the 2013 IEEE International Conference on High Performance Computing and Communications*, pp. 1553-1557.

[17] Singh, A., Rumantir, G., South, A., and Bethwaite, B. (2014). Clustering experiments on big transaction data for market segmentation. *Proceedings of the 2014 International Conference on Big Data Science and Computing*, article no.16.

[18] Tao, G., Xiangwu, D., and Yefeng, L. (2015). Parallel k-modes algorithm based on MapReduce. *Proceedings of the Third International Conference on Digital Information, Networking, and Wireless Communications*, pp. 176-179.

[19] Wang, Y., Wang, Z., and Li, X. (2004). A parallel clustering algorithm for categorical data set. *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, pp. 928-933.

[20] Yao, Z., Eklund, T., and Back, B. (2010). Using SOM-Ward clustering and predictive analytics for conducting customer segmentation. *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, pp. 639-646.

ISSN: 2304-1013 (Online); 2304-1269 (CDROM); 2414-6722 (Print)