# Clustering Algorithm for Distributed Real-Time Database sites

H. M. Abdul-kader, Rashed Salem, Safa'a Said Saleh
*Information Systems Dept*
*Menoufia University*
*Shebin Elkom*
*Eygpt*

***Abstract*— The demand for real-time database is increasing. Indeed, most real-time systems are inherently distributed in nature and need to handle data in a timely fashion. Obtaining data from remote sites may take long time making the temporal data invalid. This results in large number of tardy transactions with their catastrophic effect. Clustering the database sites nodes can help distributed real-time database systems to face the challenges meeting their time requirements. Reducing the large number of network sites into many clusters with smaller number of sites will effectively decrease the response time, resulting in better meeting of time constraints. In this paper, we introduce a clustering algorithm for distributed real-time database that depend on both the communication time cost and the timing properties of data. The results show the effectiveness of the proposed approach via achieving lower communication time, higher database performance and better meeting of timing requirements.**

***Keywords—clustering; database; real-time; distributed systems***

## I. INTRODUCTION

Recently, the demand for real-time database is increasing. Many applications such as e-commerce, mobile communication, accounting, information services, medical monitoring, nuclear reactor control, traffic control systems and telecommunications are some examples of application which require real-time data support [1].

A real-time database system (RTDBS) is defined in [2] as a database system that includes all features of traditional database system, while enforcing real-time constraints or deadlines. According to [3], the time constraints can be on the data level in the form of time validation attribute making temporal data whose validity is lost after the elapse of some pre_specified time interval, or on the transaction level in the form of deadline used by the real-time scheduling and concurrency control.

Real-time systems are often classified depending on the value of the deadline. There are three types of deadlines, *hard*, *firm* or *soft*, depending on the resulting value of the computation when missing a deadline. If the hard deadline is missed, a large or infinity penalty returns. when a firm deadline is missed, no value returns. while some value from the computation may be still for some time if the soft deadline is lost [1]. A transaction that executes outside of the deadline boundaries has less value or may damage the system, depending on the type of deadline associated with it [4].

Like any information systems, database is the main component of real-time information system. However, most real-time systems are inherently distributed in nature. Such critical systems always require to deal with their data in a timely fashion [5]. Sometimes data which is required at a particular location is not available, and it has to be obtained from remote site. This may take long time which consumes the validation duration of data make them invalid. This leads to large number of tardy transactions (transactions that miss their deadline).

In general, the number of nodes in distributed database systems has increased tremendously and the data management faces many challenges to achieve high system performance, especially in case of distributed real-time database where the transactions should be executed within specified time duration (deadline). Losing this deadline can cause a catastrophic effect. Therefore the clustering techniques which are introduced to minimize the communication time between nodes can be adapted for RTDBS to give more chance for transactions execution [6].

Clustering is a process of partitioning a given set of objects into groups of similar objects based on similarity metrics. Clustering analysis is broadly used in distributed systems researches specially to segment the network sites based on many similarity metrics to reduce the traffic load [2]. The resulting cluster can be treated as one group which is considered as a form of data compression [7].

The ultimate goal of the clustering distributed nodes is to reduce the traffic load in order to improve the network performance. Clustering techniques also are used to support the scalability which is the most important requirement for many RTDBS [8].

There are many clustering methods such as partitioning, hierarchal, density-base, modal-based. Partitioning is the most common type in which the given set of $n$ objects is partitioned into $k$ partitions, where each partition represents a cluster and $k \leq n$. Each group must contain at least one object, and in some cases each object must belong to exactly one group but this can be relaxed in some partitioning techniques [9]. Some types of partitioning methods receive some prior knowledge or physical characteristics about data to extract the optimal number of clusters with the minimum number of nodes. Determination of this knowledge represents a challenge to anyone who work in this field [10].

In fact, clustering the nodes of distributed database is useful with the scalable systems such as distributed real-time database systems that always need an effective utilization of resource [11]. This can help to avoid the un-needed communication cost between nodes during data processing [12].

However, the idea of clustering distributed database nodes according to only the physical properties as communication time as in [7] is not suitable for distributed real-time database in which the response time of transaction is very critical.

To improve the ability of distributed real-time database system to meet their strict time requirements, a novel clustering algorithm is introduced by this work to support such systems.

Our algorithm acts to segment the distributed real-time database nodes into disjoint clusters in the way to ensure that the nodes will handle data in its validation duration based on many factors including the number of shared data objects, the time properties of the shared data in addition to the communication time cost.

The remainder of this paper is organized as the following: Section II discusses the previous work that related to our work. Section III presents the proposed clustering algorithm. Section IV highlight the application of proposed algorithm in an experimental study. Section V covers the evaluation of the introduced algorithm. Finally, the conclusion is provided in section VI.

## II. RELATED WORK

Many clustering techniques have been introduced in a wide variety of applications such as image processing [2], network segmentation [13], marketing [14], pattern recognition [15], and customer segmentation[10].

Grouping database nodes into clusters with smaller number of nodes is a common clustering application in distributed systems to reduce both the communication time and the execution time so enhances the performance of the distributed database system (DDBS), specially in case of huge communication, where hundreds or thousands of node are found that increases system load and decreases system performance [7].

Different exertions to cluster the distributed databases are introduced based on different criteria. Some of them have used the clustering techniques after database fragmentation

and distributed the resulted fragments over the clusters not over database nodes. This could speed up the data allocation process by eliminating the extra communication costs between distributed database nodes as in [12, 16, 17]. Such clustering methods were based on predefining number of clusters.

Other efforts are introduced to cluster the networked nodes of DDBS such as the work by Srikanth et al. [18] which considered the energy efficiency of WSN to segment the nodes. Their algorithms performed better in terms of network life time, energy consumption and number of dead sensor nodes. Another example by Abbasia et al. [8] which based on the network architecture and aims to enable energy effect network operation as in case of sensor wireless network, others are based on the properties of the generated clusters.

A wide section of clustering techniques are based on the communication cost (time in ms/byte) between DDBS to segment the nodes according to it as in [19] and the work by Hababeh [7] which is the most related work to our proposed algorithm. The author introduced a clustering technique that divided the database network nodes into disjoint clusters based on the smallest average communication cost between the database nodes. This can achieve a better network distribution, and minimize communication time between database nodes. Therefore a higher performance of distributed database system was recognized.

Bearing in mind, the nature of time-constraint database, we will still need to develop a novel clustering algorithm to outperform the current clustering techniques. The novel algorithm requires to create disjoint clusters according to the properties of data and network together to generate near optimal number of clusters required. Moreover, the proposed algorithm supports the real-time databases to meet the timing requirements and avoid the fetal effect of missing the deadline in case of hard transactions, thus improving the DDBS performance.

Our proposed work is designed to benefit from the idea of the work by Hababeh [7] to reduce the communication traffic and improve the performance of DRTDB by extending this clustering algorithm to include the timing features of RTDB in clustering criteria in addition to the network properties.

## III. THE PROPOSED ALGORITHM

The main contribution of this paper is to introduce an efficient clustering algorithm for distributed real-time database that may be based on any database model and built on any network topology. This clustering algorithm should act to map between the physical properties of network and the timing properties of data to ensure that the transferred data will reach in their validation duration to the requested site.

### *A. The Clustering Parameters*

The following list define and describe the parameters of the proposed algorithm:

- Total Number of Nodes(N)

- Total Number of real-time Objects (O)
- Communication Cost matrix CC(Si, Sj):The cost of creating and transmitting database object in ms/byte between any two nodes $S_i$ and $S_j$ in the distributed database network system.
- Log matrix (LM): The working set that identify the objects that used by all transactions at each node. It is obtained from a customize log file about the data object with time constraint properties in each node S(i).
- Deadline[O][N]: a source of minimum deadline or validity duration of the transaction on each node
- Cluster set matrix (CSM[N][O][N]) : a place to identify the node cluster of each object on each node
- Total shared objects matrix (N2N[N][N]) :Total number of shared objects between each two site
- Cluster decision value (CDV): The value that determine if the two node can be grouped together in one cluster or not. This value can be identified as seen in "1" based on the rule that the nodes that share number of objects equal to or greater than the threshold(4 in our case study).

$$\text{CDV}(S_i, S_j) \begin{cases} 1 \; N2N(Si, Sj) \geq 4 \;\; \wedge^{i \neq j} \\ 0 \; N2N(Si, Sj) < 4 \;\; \vee^{i=j} \end{cases} \qquad (1)$$

*B. The Clustering algorithm*

The decision of clustering is based mainly on the rule that the communication cost between two sites will never affect the validity of transmitted data , that is the data object must be transmitted within its validation period. The number of allowed data objects between two sites is another clustering criteria.

To maintain and simplify the work, these assumption are considered:

1. The communication cost between each two sites are symmetric.
2. The communication cost in the same site is zero.
3. The transaction type is hard

**Inputs**

Log matrix (LM), Communication Cost matrix CC[N][N] , Total Number of Nodes (N) Total Number of data Objects (O)

**Processing**

{Module 1: Obtaining Min Deadline Matrix}
{Module 2: Comparing CC with Minimum Deadline}
{Module 3: Obtaining total shared data Objects}
{Module 4: Determine the cluster for each site &each data object}

**Output:**

Matrix of generated clusters for each site (CSM[][][])

*1) Module 1: Obtaining Min Deadline*

This module checks the transaction deadline (DL) and Validity Duration (VD) for each transaction and exploited data in S[i] to produce Minimum Time matrix.

step1: For each row in LM
step2: deadline[objects][Nodes]= min (VD,DL)
step 3: loop

*2) Module 2: Comparing CC with Minimum Time*

This module determines the sites that match the minimum time cost which is less than the minimum deadline of shared data object between them in order to group them initially in one cluster.

step 1: Do for each row in deadline[objects][Nodes] using counter i
step 2: set ObjectID = i; and the counter C = 0
step 3: set Boolean matrix: CSM[nodes][objects][nodes]to false;
step 4: Do for each col in deadline[o][N] using counter j
step 5 : If (deadline[i][j] ≠ 0) then
set VD= deadline[i][j]; set NodeID =j;
set temp[C++]=NodeID; // temporary array,
step 6: Loop
step 7: Do using counter x from 0 to C-2
step 8: Do using counter y from x+1 to C-1
step 9 : if CC[temp[x]][temp[y]]<VD then in
set CSM[temp[x]][ObjectID][temp[y]] = true;

set CSM[temp[y]][ObjectID][temp[x]] = true;
end if
step 10 : loop
step 11: loop
step 12: loop

*3) Module 3: Obtaining total shard data Objects*

This module checks the clustering set matrix (CSM) for all nodes to produce the total number of shared data object between each two sites that can be transmitted between them while their validity.

step 1: Do for each row in CSM of node x = 1 to N
step 2: set counter[] =0;
step 3: Do for counter i = 1 to N
step 4 : Do for counter j = 1 to N
step 5 : If CSM[i][j] = true then counter[j]++
step 6: Loop
step 7: Loop
step 8: DO : for counter z = 1 to N
N2N[x][z] = counter[z]
step 9: Loop

*4) Module 4: Determine the cluster for each site &each data object*

This module distribute the DRTDB nodes into clusters according to the computations performed in Module 2 and Module 3

step 1:set x = 1
step 2:set first cluster cluster[x] = null
step 3:Dofor i = 1 to 1 // for the first node
step 4: add nodeID= i to cluster[x]
step 5: Do for j= 1 to N
step 6: if i j then
step 7: if N2N[i][j] the threshold then add j to cluster[x] else add to cluster_w[x]end if
step8: end if
step9: Loop
step 10: Loop
step 11: Do for i = 2 to N
step 12: Do for j= 1 to N
step 13: if i j then
step 14: Do for y = 1 to x
step 15:if i and j are found in cluster[y]or in cluster_w[y] then continue
step 16: else if N2N[i][j] ≥ thresh AND N2N[i][j] ≠then add i & j into cluster[++x]
step 17: else add i& j into cluster_w[++x]end if
step19: end if
step 20: Loop
step 21: end if
step 22: Loop
step 23: Loop

## IV. EXPERIMENTAL STUDY

Our experimental clustering analyses were conducted over the DDBS sites using the proposed clustering method. The results obtained from the experimental tests confirm that our approach can be implemented in different DDBS environments even for the network with larger number of sites.

For simplicity, we build a complete connected network consists of 10 nodes supported by a full-distributed database over different areas. These nodes contain 100 data objects with time constraint properties on data level with validity duration between (1-2.8 sec). For simplicity again, each data object contains only one time-constraint attribute.

All nodes are used as servers that execute and develop the database transactions, and also all of them are used as terminals which produce administrational and functional information. The communication time cost between sites in ms/byte is presented in Table 1.

TABLE 1. COMMUNICATIONS COST MATRIX : CC[N][N]

| | Node (1) | Node (2) | Node (3) | Node (4) | Node (5) | Node (6) | Node (7) | Node (8) | Node (9) | Node (10) |
|---|---|---|---|---|---|---|---|---|---|---|
| NODE 1 | 0 | 0.1 | 1.1 | 0.6 | 0.4 | 0.8 | 1.5 | 1.6 | 0.4 | 0.5 |
| NODE 2 | 0.1 | 0 | 1 | 1 | 0.2 | 0.3 | 0.4 | 0.2 | 0.7 | 0.6 |
| NODE 3 | 1.1 | 1 | 0 | 0.6 | 0.6 | 0.7 | 0.8 | 0.7 | 0.8 | 0.2 |
| NODE 4 | 0.6 | 1 | 0.6 | 0 | 0.8 | 0.7 | 0.6 | 0.7 | 0.1 | 0.1 |
| NODE 5 | 0.4 | 0.2 | 0.6 | 0.8 | 0 | 0.1 | 0.2 | 0.2 | 0.7 | 0.9 |
| NODE 6 | 0.8 | 0.3 | 0.7 | 0.7 | 0.1 | 0 | 0.1 | 0.2 | 0.6 | 0.8 |
| NODE 7 | 1.5 | 0.4 | 0.8 | 0.6 | 0.2 | 0.1 | 0 | 0.4 | 0.7 | 0.6 |
| NODE 8 | 1.6 | 0.2 | 0.7 | 0.7 | 0.2 | 0.2 | 0.4 | 0 | 0.8 | 0.7 |
| NODE 9 | 0.4 | 0.7 | 0.8 | 0.1 | 0.7 | 0.6 | 0.7 | 0.8 | 0 | 0.1 |
| NODE 10 | 0.5 | 0.6 | 0.2 | 0.1 | 0.9 | 0.8 | 0.6 | 0.7 | 0.1 | 0 |

A customized transaction log file is generated and collected from all node into one matrix LM. Fig. 1 show a sample of the generated log file matrix that is needed by module1.

logMaterix

| Node | objectID | duration |
|---|---|---|
| 2 | 27 | 2.2 |
| 2 | 28 | 2.2 |
| 2 | 34 | 2.8 |
| 2 | 54 | 2 |
| 2 | 59 | 1.71 |
| 2 | 64 | 1 |
| 2 | 80 | 1.45 |
| 2 | 89 | 1 |
| 3 | 2 | 2 |
| 3 | 2 | 2 |
| 3 | 3 | 1.5 |
| 3 | 3 | 1.5 |
| 3 | 4 | 1.2 |
| 3 | 5 | 1.7 |
| 3 | 5 | 1.7 |
| 3 | 6 | 1.7 |
| 3 | 6 | 1.7 |
| 3 | 8 | 1.1 |
| 3 | 9 | 1.1 |
| 3 | 10 | 1.8 |
| 3 | 14 | 2 |
| 3 | 20 | 1.13 |
| 3 | 21 | 1.13 |
| 3 | 26 | 2.1 |
| 3 | 38 | 2 |

Fig. 1. Sample of Log Matrix: LM

Processing the module #1 results in the deadline matrix. This matrix determined the nodes which handle each real-time data object and the minimum deadline of each. Table 2 presents a sample of this matrix.

The module #2 used this Deadline matrix to determine the nodes which deal with each data object and store them in a temporary array. Fig. 2 show a sample of it.


Fig. 2. Temp array Sample : temp[C]

TABLE 2. SAMPLE OF DEADLINE MATRIX

| objectID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 13 | | | | | 1 | | 1 | 1 | | |
| 14 | | | 2 | 2 | | | | | 2 | 2 |
| 15 | | | | | 1.6 | | | 1.6 | 1.6 | |
| 16 | | | | | 1.8 | 1.8 | | 1.8 | | |
| 17 | | | | | | 1.8 | 1.8 | | | 1.8 |
| 18 | | 1.14 | | | 1.14 | | | 1.14 | | |
| 19 | | 1.14 | | | | 1.14 | | 1.14 | | |
| 20 | | | 1.13 | | 1.13 | | 1.13 | 1.13 | | |
| 21 | | | 1.13 | | 1.13 | 1.13 | | 1.13 | | |
| 22 | | | | | 2 | | 2 | | | |
| 23 | | | | | | 2 | | 2 | | |
| 24 | | | | | | 2 | | 2 | | |
| 25 | | | | | | | 2 | 2 | 2 | |
| 26 | | 2.1 | 2.1 | 2.1 | | | | | | |
| 27 | | 2.2 | | | 2.2 | 2.2 | | | | |
| 28 | | 2.2 | | 2.2 | | 2.2 | 2.2 | 2.2 | | |
| 29 | | | | 2.5 | 2.5 | | 2.5 | 2.5 | | |
| 30 | 2.5 | | | | 2.5 | 2.5 | | 2.5 | | |
| 31 | | | | | | 2.7 | | 2.7 | | |
| 32 | | | | 2.7 | 2.7 | | 2.7 | 2.7 | | |
| 33 | | | | | | 2.7 | 2.7 | 2.7 | 2.7 | |
| 34 | 2.8 | 2.8 | | 2.8 | 2.8 | | | | | |
| 35 | 2.8 | | | | | | | | | |
| 36 | | | | 2.8 | | 2.8 | | 2.8 | | |
| 37 | 2.8 | | | 2.8 | 2.8 | 2.8 | | 2.8 | | |

These arrays are used later by the same module to compare the communication time cost versus the deadline property of the real-time data objects. This make the first decision that identify the nodes which can transmit the data object to them within its validation period. That is what we called CSM matrix, Fig. 3 shows a sample of them.

NOD1

| ID | NOD2 | NOD3 | NOD4 | NOD5 | NOD6 | NOD7 | NOD8 | NOD9 | NOD10 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | T | | T | | | T | | T | |
| 3 | T | T | | | | FALSE | | | |
| 30 | | | | T | T | | T | | |
| 34 | T | | T | T | | | | | |
| 35 | | | | | | | | | |
| 37 | | | T | T | T | | T | | |
| 4 | T | T | | | T | FALSE | | T | |
| 42 | | | | T | | FALSE | | | |
| 43 | | | T | | T | | | T | |
| 47 | | | T | | | | | | |
| 48 | | | | | T | | | | T |
| 64 | T | | | T | | | | | |
| 65 | | FALSE | | | T | | | | |
| 7 | | | | T | | FALSE | FALSE | | |
| 86 | | | | | | | | T | |
| 90 | | FALSE | | | T | | | | |
| 92 | | | | | T | | FALSE | | |

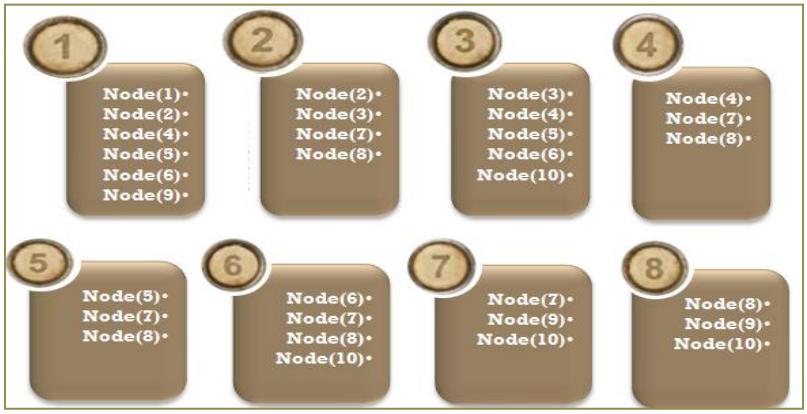
Fig. 3. samples of resulted CSM for node 1

Clusters can be generated by performing the third module using the following matrix in Table 3 that summarizes the total number of data objects that can be allowed to share between each two nodes.

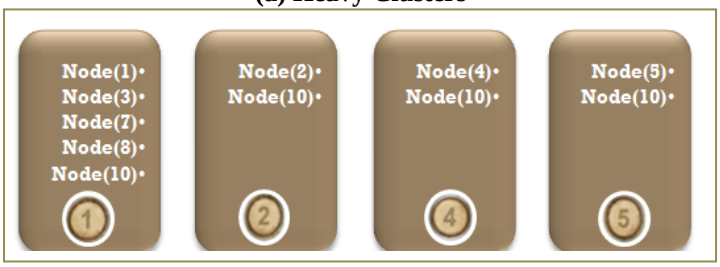
TABLE 3. TOTAL SHARED OBJECTS MATRIX (N2N[N][N])

| node | node 1 | node 2 | node 3 | node 4 | node 5 | node 6 | node 7 | node 8 | node 9 | node 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 5 | 2 | 5 | 6 | 8 | 1 | 2 | 4 | 1 |
| **2** | 5 | 0 | 6 | 5 | 4 | 7 | 6 | 5 | 4 | 2 |
| **3** | 2 | 6 | 0 | 7 | 8 | 6 | 7 | 5 | 7 | 4 |
| **4** | 5 | 5 | 7 | 0 | 6 | 7 | 6 | 7 | 6 | 3 |
| **5** | 6 | 4 | 8 | 6 | 0 | 9 | 10 | 14 | 3 | 3 |
| **6** | 8 | 7 | 6 | 7 | 9 | 0 | 10 | 16 | 11 | 7 |
| **7** | 1 | 6 | 7 | 6 | 10 | 10 | 0 | 16 | 9 | 5 |
| **8** | 2 | 5 | 5 | 7 | 14 | 16 | 16 | 0 | 10 | 6 |
| **9** | 4 | 4 | 7 | 6 | 3 | 11 | 9 | 10 | 0 | 16 |
| **10** | 1 | 2 | 4 | 3 | 3 | 7 | 5 | 6 | 16 | 0 |

Part of resulted clusters from our algorithm in two categories is presented in Fig. 4. Heavy clusters with prefix HC those contain nodes that share number of objects is equal to or higher than the threshold. However, light clusters are those that contain nodes that share number of objects is less than the threshold.




**(a) Heavy Clusters**




**(b) Light Clusters**

Fig. 4. The resulted clusters

## V. CLUSTERING EVALUATION

Table 4. presents statistics about the resulted clusters from the proposed algorithm. Notice that the total communication cost between all sites is 594 ms (calculated from table1) while highest total communication after our algorithm is 206 ms, that means; the introduced algorithm can reduce around 65% from the total communication cost with communication average with 0.88 in each communication.

TABLE 4. THE RESULTED CLUSTERS STATISTICS

| CLUSTER | HC1 | HC2 | HC3 | HC4 | HC5 | HC6 | HC7 | HC8 | LC1 | LC2 | LC4 | LC5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | 6 | 4 | 6 | 3 | 3 | 4 | 3 | 3 | 5 | 2 | 2 | 2 |
| CC | 180 | 90 | 206 | 58 | 80 | 120 | 60 | 64 | 98 | 8 | 6 | 6 |
| CCR | 414 | 504 | 388 | 536 | 514 | 474 | 534 | 530 | 496 | 586 | 588 | 588 |

N: number of nodes; CC: Total communication cost; CCR: reduced communicationcost

On the other hand, Hababeh in [20] used the clustering performance evaluation (CPE) to evaluate his distributed database clustering algorithm. He defined CPE as the result of the reduced costs of communications divided by the sum of communication costs between sites. The reduced communication costs was computed as the difference between the sum of costs that are required for each site to communicate with other sites in the network system and the sum of costs that are required for each cluster to communicate with other clusters.

$$CPE = ((\sum_{i=1}^{n}\sum_{j=1}^{n} CC(S_i, Sj) - (\sum_{i=1}^{n}\sum_{j=1}^{n} CC(C_i, Cj)))/(\sum_{i=1}^{n}\sum_{j=1}^{n} CC(S_i, Sj)) \quad (2)$$

CC(*Si*,*Sj*): communication cost between any two nodes Si and Sj in the cluster $C_i$ where i,j = 1,2,3, …, n.
CC($C_i$,$C_j$): communication cost between any two clusters; $C_i$ and $C_j$ where i,j = 1,2,3, …, n

But the work by Hababeh in [20] generated disjoint groups of network sites according to their communication costs and clustering range. They prevented the overlapping between clusters and allowed the clusters interconnection.

In term of CPE, our proposed algorithm obtained CEP higher than that can be obtained from his work as we prevent the communication between clusters that make the value of CEP equals 1, while this value must be less than one in his work, because he allowed the communication between clusters.

- $\sum_{i=1}^{n}\sum_{j=1}^{n} CC(S_i, S_j)$ = 594
- $\sum_{i=1}^{n}\sum_{j=1}^{n} CC(C_i, C_j)$ = 0
- *CPE = 1*

The number of clusters obtained is another criteria to evaluate the clustering algorithm. Hababeh in [20]compared his results with the studies in [21] and [22] as they were the most closest to his clustering technique. The author in the work [21] defined a basic parameter to identify the clustering. Also they allow object overlapping between clusters. In contrast, the author in [22] used the natural logarithm function to determine the approximation of the average cheapest path length between network nodes, as more network sites added, more clusters are generated due to the nature of logarithmic function used. The resulted number of clusters from [20] , [21]and [22] are compared with the proposed clustering technique results. Fig. 5 depicts the clustering performance evaluation of these methods under comparison in term of the number of clusters. As we can see more clusters are generated by our clustering method. On the other hand, less number of clusters is generated from [20], [21] and [22] .

On the other hand, the proposed clustering algorithm use the properties of network communication with the properties of distributed data to generate disjoint clusters allowing the overlapping and preventing the communication between clusters to support time meeting requirements in real-time database.

As the most related work by Hababeh in [20], more clusters are generated as the number of sites increase with a normal distribution in the network system.

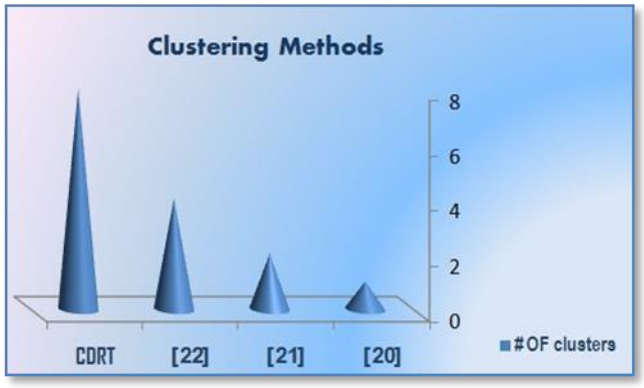



Fig. 5. Clustering Performance evaluation

## VI. CONCLUSIONS

This work has introduced a novel clustering algorithm for distributed real-time database that act to map between the network communication time cost and the timing properties of the distributed data. We showed that our clustering technique is able to group the distributed database system network sites into number of clusters and reduce the communication overhead between the sites. As consequence, this reduction must enhance the performance and increase the chance that the distributed database network system can meet critical time- requirements. Reducing the large number of network sites into many clusters with smaller number of sites will effectively decrease the response time, resulting in better meeting of time constraints. Considering this clustering technique as a pre-step of the data replication process will improve the distributed database system.

We planned to treat the problem of false allowed sites in CSM(two sites that share in data but the communication cost between them is larger than the validation duration property of the shared data) by searching an indirect path that may has cheaper cost and allow two sites to exchange data in its validation time within a cluster( i.e. indirect cost is smaller than deadline). This may need to identify new data structure that can combine simple and complex paths.

## REFERENCES

[1] Vishnu S., Gyanendra K. G., UdaiSh.r; Issues In Mobile Distributed Real Time Databases: Performance And Review; International Journal of Engineering Science and Technology (IJEST);2011

[2] Aslinger. A., Sang H. Son; Efficient Replication Control in Distributed Real-Time Databases; IEEE;2005.

[3] Deysy G., Hasdai P., Andrea M.; Analysis of Clustering Algorithms for Image Segmentation and Numerical Databases; IEEE;2008

[4] Andrew S.T., Maarten V. S.; DISTRIBUTED SYSTEMS principals and paradigm; Pearson Prentice Hall; 2008.

[5] Younis O, Fahmy S; Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach. In: The conference on computer communications; the twenty-third conference of the IEEE communications societ; March 2004

[6] Hababeh I, Ramachandran M, Bowring; A high-performance computing method for data allocation in distributed database systems; J Supercomput; 2007.

[7] Hababeh I;Improving network systems performance by clustering distributed database sites; J Supercomput (2012); © Springer Science+ Business Media.

[8] Sanny G., Sten F. A.; Decentralized and Continuous Consistency Management in Distributed Real-Time Databases with Multiple Writers of Replicated Data; IEEE; 2005.

[9] AndlerS., Hansson J., Eriksson J., MellinJ., BerndtssonM., and EftringB.; DeeDS towards a distributed and active real-time database system; SIGMOD Record, 1996.

[10] AbbasiaA. A., YounisM.; A survey on clustering algorithms for wireless sensor networks; Published by Elsevier B.V.; 2007

[11] LingrasP. and Wes C.; "Interval set clustering of web users with rough k-means"; Journal of Intelligent Information Systems; 2004

[12] ShyuM., Chen S., and Rubin S.; "Stochastic Clustering for Organizing Distributed Information Sources"; IEEE Transactions on Systems, Man, And Cybernetics-Part B:Cybernetics; 2004,

[13] WeiWang, Shidong; Application of Data Mining Technique in Customer segmentation of Shipping Enterprises; IEEE,2010

[14] Fragkiskos D. Malliarosa, Michalis V.; Clustering and community detection in directed networks: A survey; Elsevier; 2013.

[15] Jiawei H., Micheline K.; Data Mining: Concepts and Techniques, Second Edition; Elsevier Inc ; 2006

[16] Hababeh I, Ramachandran M, Bowring N; Designing a high performance integrated strategy for secured distributed database systems. Int J Comput Res (IJCR); 2008

[17] Son J, Kim M; An adaptable vertical partitioning method in distributed systems; 2004.

[18] Ma H, Scchewe K, Wang Q; Distribution design for higher-order data models. Data KnowlEng; 2007.

[19] Srikanth J., Prasanta K.; Energy Efficient Grid Based Clustering and Routing Algorithms for Wireless Sensor Networks Department of Computer Science & Engineering; Fourth International Conference on Communication Systems and Network Technologies ; 2014.

[20] Hababeh I., Intelligent Network Communications for Distributed Database Systems; Second International Conference on Advances in Databases, Knowledge, and Data Applications; IEEE computer society; 2010.

[21] Kumar P, Krishna P, Bapi R, Kumar S; Rough clustering of sequential data. Data KnowlEng; 2007.

[22] Fronczak A, Holyst J, Jedyank M, Sienkiewicz J; Higher order clustering coefficients. Barabasi Albert networks. Physica; 2002.