Balanced Workload Clusters for Distributed Object Oriented Software

Heba Ragab[1], Amany Sarhan[1], Al Sayed Sallam[1], and Reda Ammar[2]
[1]Computer and Control Engineering Department, Tanta University, Egypt
[2]Computer Science and Engineering Department, University of Connecticut, USA

Abstract: *When clustering objects to be allocated on a number of nodes, most researches focus only on either the communication cost between clusters or the balancing of the workload on the nodes. Load balancing is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives or other resources, in order to, get optimal resource utilization, maximize throughput, minimize response time and avoid overload. In this paper, we introduce three clustering algorithms that obtain balanced clusters for homogeneous clustered with minimized communication cost.*

Keywords: *Load balance, distributed system, software restructuring, cluster algorithms.*

Received June 28, 2012; accepted July 28, 2013; published online August 17, 2014

1. Introduction

Understanding the functionality of large software systems is a difficult job because of their inherent complexity. So, the system can be broken down into smaller pieces that are easier to comprehend and to manipulate. Software clustering is a very important facility since, it helps identify the subsystems that have related functionality and are somehow independent from the other parts of the system. Therefore, the starting point in the process of distributing the objects (or tasks) is to identify the clusters that constitute the software system [31]. The interdisciplinary nature of clustering is evident through its vast literature which includes many clustering problem formulations and even more algorithms. Basically, the two main approaches to clustering are hierarchical clustering and partitioning clustering [24, 25]. A hierarchical clustering is necessary here to model the distinct abstraction levels of the object oriented software [1]. The hierarchical clustering allows the designer to overcome the complexity of modeling a large application with multiple levels of abstraction and large numbers of interacting objects. This modeling framework helps the system designer to derive the information required to construct a software application that meets a set of performance requirements. In a Distributed Object Oriented (DOO) application, method invocation is the only communication pattern between objects within an application. An important step in designing DOO systems is to decide upon object locations. There are two main issues concerning clustered architectures. The first one is the communication between clusters. The second issue is the workload balance. Both of them depend greatly on the technique used to distribute the program instructions. This distribution should minimize the communication needs between remote locations [7]. However, it may be necessary to use tools to allow analysis of the communication patterns among objects in order to, take the right decision. However, it is scarcely to link together a collection of distributed and cost-effective in the form of a cluster [6]. The Class Dependency Graph (CDG), as shown in Figure 1 is usually used to create a suggested grouping of subsystems that are convenient for guiding the allocation of the subsystems to the targeted distributed environment [24, 33, 37].

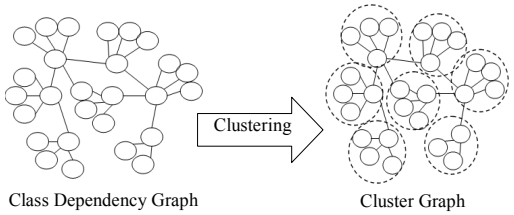


Figure 1. The clustering step of the restructuring approach.

Besides that, it is also important to obtain even clusters with equal, or nearly equal, computation requirements. Workload is referred to as the amount of processing that the computer has been given to perform at a given time. The workload consists of some amount of programming running in the computer and usually some number of users connected to and interacting with the computer's applications [34].

Load balancing is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to, get optimal resource utilization, maximize throughput, minimize response time and avoid overload. The problem of load balancing continues to raise interesting challenges to researchers. The operating systems and distributed application design must include solutions for solving it. Cluster analysis has been the most popular statistical technique for dividing the workload into workload classes. Workload management enables workload distribution to provide

optimal performance for users and applications. Workload transfer is done once the node fails as the load balancer connects to the nodes works like a pulse that monitors the performance of each node. With a predetermined interval, the load balancer will check each node and alert the network administrator in case one node fails.

Balancers in load balancing cluster have the ability to transfer the workload to other nodes to avoid further delay of operation [6]. Clustering pushes the functions of an application to be faster or ensures data availability's faster transfer. An ideal clustering form to ensure stability is load balancing. Clustering by load balancing is basically a form of connection between computers (referred to as nodes in clustering) where in the workload is evenly distributed. Although, a single computer could provide the same operations, the hardware capability of a single computer or a node will never be enough to handle massive data requests and processing. However, multiple computers are connected together to harness each processing power. By sharing the workload, the processing time is increased and massive data requests and processes could be possible [21, 28].

Load balancing clusters provide a more practical system for business needs. As the name implies, that system entails sharing the processing load as evenly as possible across a cluster of computers. The differentiating factor in this case is the lack of a single parallel program that runs across these nodes. Each node server in that type of cluster, in most cases, is an independent system running separate software. However, there is a common relationship between the nodes either in the form of direct communications between the nodes or through a central load balancing server that controls each node's load. Usually, a specific algorithm is used to distribute that load. This load could be in the form on application load or network load that needs to be balanced [35].

In this paper, we introduce three algorithms for clustering the objects into number of clusters to match the existing hardware named: Hierarchical and K-Partitioning, Hierarchical and K-Medoids and Double K-Medoids. These algorithms aim to cluster the distributed objects into present numbers of clusters considering two main issues: Minimizing the communication between clusters and achieving the workload balance of the clusters. These algorithms, through simulation results, proved to obtain better performance than the existing clustering algorithms.

This paper is organized as follows. This section provides the introduction. Section 2 describes the previous clustering techniques. Section 3 includes the three proposed algorithms for clustering distributed objects. Section 4 discusses the system cost model; communication and workload costs. Simulation results and their analysis are covered in section 5. The conclusion of the paper summarizes the work presented. Finally, a list of references used in the research is given.

2. Previous Work in Clustering

2.1. K-Partition Algorithm

K-Partition (or K-means) algorithm is based on an idea to obtain k clusters, split the set of all points into two clusters, select one of these clusters to split the set of points into two clusters, select one of these clusters to split and so on, until k clusters have been produced. Therefore, all points in a given subset are closest to the same center. The calculated cluster center V_i ($i \in \{1, 2, ..., K\}$) is the mean of the objects points in cluster i. K-Means cluster analysis uses Euclidean distance to compute the distances. The K-Partition algorithm has the following important properties:

- It is efficient in processing large objects sets.
- It often terminates at a local optimum.
- The clusters have spherical shapes.
- It is sensitive to noise.

The main problem of K-Means algorithm is the random initialization of centers [2, 3, 4, 10, 11, 22, 24, 27]. Following is the summary of the K-Partition algorithm.

Algorithm 1: K-Means algorithm

Given the objects set X and the number of clusters C, 1< C< N.
Initialize with random cluster centers chosen from the objects set.
Repeat for iteration l=1, 2,…

Step 1: Compute the square distances N is the number of object points.

$$D_k=(x_k- v_i)^T (x_k- v_i)$$

Step 2: Select the points for a cluster with minimal distances belong to the cluster.
Step 3: Calculate cluster centers.

until

$$V_i^{(l)} = \frac{\sum_{j=1}^{N_i} x_i}{N_i}$$

$$\prod_{k=1}^{n} max \left| V^{(l)} - V^{(i-1)} \right| \prime 0$$

end

2.2. K-Medoids Algorithm

K-Medoids and K-Partition algorithms both attempt to minimize squared error, the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to K-Partition, K-Medoids choose object points as centers. The main difference between K-Partition and K-Medoids stands in calculating the cluster centers: The new cluster center is the nearest object point to the mean of the cluster points.

Algorithm 2: K- Medoids algorithm

Given the objects set X and choose the number of clusters C, 1< C< N.
Initialize with random cluster centers chosen from the objects set X.
Repeat for iteration l=1, 2, …

Step 1: Compute the square distances.

$$D_k^2 = (x_k - v_i)^T (x_k - v_i) \quad Where, 1 \le k \le c, 1 \le k \le N$$

Step 2: Select the points for a cluster with minimal distances belong to the cluster.

Step 3: Calculate cluster centers.

$$V_i^{(1)} = \frac{\sum_{j=1}^{N_i} x_i}{N_i}$$

Step 4: Choose the nearest objects point to be the cluster center.

$$D_{ik}^{2*} = (x_k - v^*_i)^T (x_k - v^*_i) \text{ and}$$

$$x^*_i = argmin(D_{ik}^{2*}); v_i^{(1)} = x^*_i$$

Until $\prod_{k=1}^{n} max b \left| V^{(1)} - V^{(i-1)} \right| \prime 0$

End.

2.3. Double K-Clustering (D-K Partition) Algorithm

D-K Clustering algorithm uses the K-Partitioning algorithm twice. In the first time, the original CDG will be clustered according to the number suggested by the recursive clustering algorithm. In the second time, the K-Partitioning algorithm will be used again to group the resultant clusters and form the MCG. These steps are illustrated in Figure 2 [17, 18].

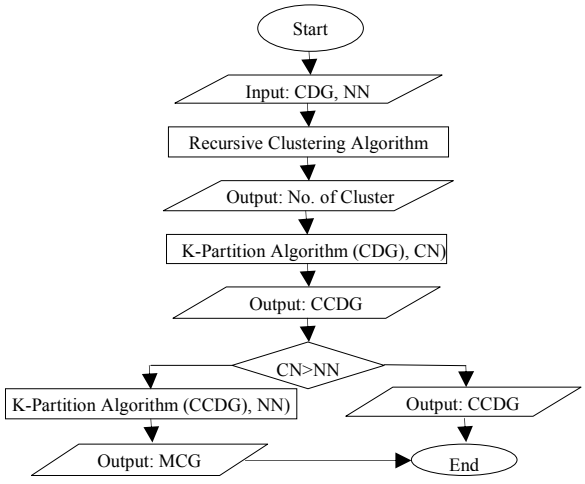


Figure 2. The steps of the D-K Partition Algorithm.

2.4. Hierarchical Algorithm

Hierarchical clustering techniques belong to a second category of clustering methods, however used as a method for grouping different signals. Hierarchical clustering builds a cluster hierarchy or in other words, a tree of clusters, also known as a dendrogram. Investigation based on Euclidian distance measures.

Strategies for hierarchical clustering generally fall into two types:

- Agglomerative: This is a "bottom-up" approach where each observation starts in its own cluster and pairs of clusters are merged as one move up the hierarchy.
- Divisive: This is a "top-down" approach where all observations start in one cluster and splits are performed recursively as one move down the hierarchy.

Both methods suffer from their inability to perform adjustments once the splitting or merging decision is made. Advantages of hierarchical clustering include:

- Embedded flexibility regarding the level of granularity.
- Ease of handling of any forms of similarity or distance.
- Consequently, applicability to any attributes types.

Hierarchical clustering initializes a cluster system as a set of singleton clusters (Agglomerative case) or a single cluster of all objects (Divisive case) and proceeds iteratively with merging or splitting of the most appropriate clusters until the stopping criterion is achieved. The appropriateness of clusters for merging/ splitting depends on the similarity/ dissimilarity of clusters elements.

To merge or split subsets of points rather than individual points, the distance between individual points has to be generalized to the distance between subsets. Such derived proximity measure is called a linkage metric. The type of the linkage metric used significantly affects hierarchical algorithms, since, it reflects the particular concept of closeness and connectivity [3, 4, 5, 8, 9, 15, 19, 22, 24, 25, 26].

2.4.1. Agglomerative Hierarchical Method

Agglomerative hierarchical techniques are starting with individual objects as clusters; merge the two closest clusters until only one cluster remains. There are three definitions of the closeness between two clusters: Single-link, complete-link and average-link.

The single-link similarity between two clusters is the similarity between the two most similar instances, one of which appears in each cluster. The complete-link similarity is the similarity between the two most dissimilar instances, one from each cluster. The average-link similarity is a compromise between the two [5, 9, 24].

Algorithm 3: Agglomerative hierarchical method

Step 1: Compute the linkage metrics.
Step 2: Merge the closest two clusters.
Step 3: Update the linkage metrics.
Step 4: Repeat.
Step 5: Until only one cluster remains.

Agglomerative methods start with individual objects and group them together to form larger and larger classes. The algorithm terminates when all objects are combined to one class. At every stage one wants the two similar classes to be amalgamated. The most established amalgamated techniques viewed as special cases of the general agglomerative algorithm. The dissimilarity between newly amalgamated class measures:

$$d(C_i{}^*C_j, C_k) = a(i)d(C_i, C_k) + a(k)d(C_j, C_k) + bd(C_i, C_k) + c \mid d(C_i, C_k) - d(C_j, C_k) \mid \quad (1)$$

Here a, b and c are coefficients corresponding to a particular linkage. This formula expresses a linkage metric between the union of the two clusters and the third cluster in terms of underlying components. The Lance-Williams formula has an utmost importance since, it makes manipulation with dis(similarity) computationally feasible.

Linkage metrics-based hierarchical clustering suffers from time complexity. Under reasonable assumptions, such as reducibility condition (graph methods satisfy this condition), linkage metrics methods have $O(N^2)$ complexity. Despite the unfavorable time complexity, these algorithms are widely used [3, 4, 7, 15].

3. The Proposed Algorithms

3.1. Hierarchical and K-Partitioning (H-K Partition) Algorithm

The most popular algorithms are Hierarchical algorithm and K-Partitioning algorithm. We present a hybrid approach to combine the merits of the two classic approaches and discard disadvantages. Hierarchical clustering builds a cluster hierarchy or, in other words, a tree of clusters, also known as a dendrogram. Investigation is based on Euclidian distance measures.

The dendrogram will be the input to the K-Partition algorithm which will cause splits moving down the dendrogram. The K-Partition algorithm will perform Splits Cluster Graph (SCG) at first. However, in some cases we need to Merge the Cluster Graph (MCG) to determine the number of available nodes. As shown in Figure 3, we defined this number of nodes as NN and cluster nodes as CN. The output of this step will be groups whose numbers equal to the number of available clusters in the distributed system.

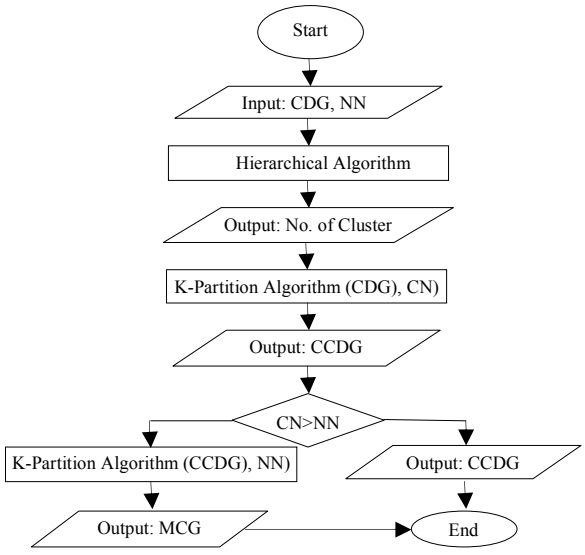


Figure 3. The steps of the H-K Partitioning algorithm.

3.2. Hierarchical and K-Mediods (H-K Medoids) Algorithm

The second hybrid approach is based on the K-Mediods algorithm rather than the K-Partitioning. Hierarchical algorithm is used as first step to prepare the dendrogram. Investigation is based on Euclidian distance measures. The dendrogram will be the input to the K-Mediods algorithm which will cause splits moving down the dendrogram. The next step is K-Medoids algorithms.

As shown in Figure 4, we defined this number of nodes as NN and cluster nodes as CN. The output of this step will be groups whose numbers equal to the number of available clusters in the distributed system.

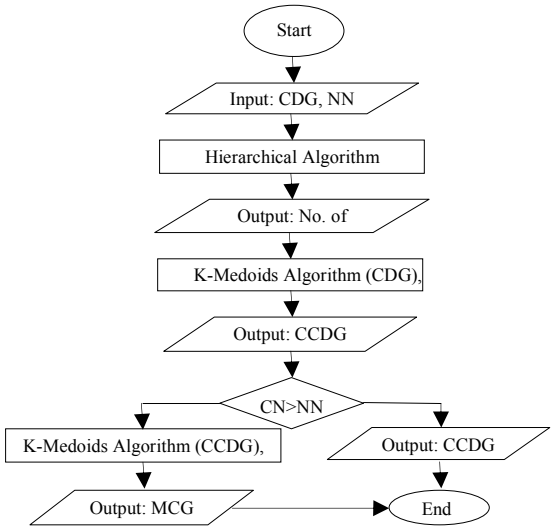


Figure 4. The steps of the H-K Medoids algorithm.

3.3. Double K-Medoids

Double K-Medoids algorithm is using the K-Medoids algorithm twice. In the first time, the original CDG will be clustered according to the number suggested by the recursive clustering algorithm. In the second step, the K-Medoids algorithm will be used again to group the resultant clusters and to form the MCG. These steps are illustrated in Figure 5.

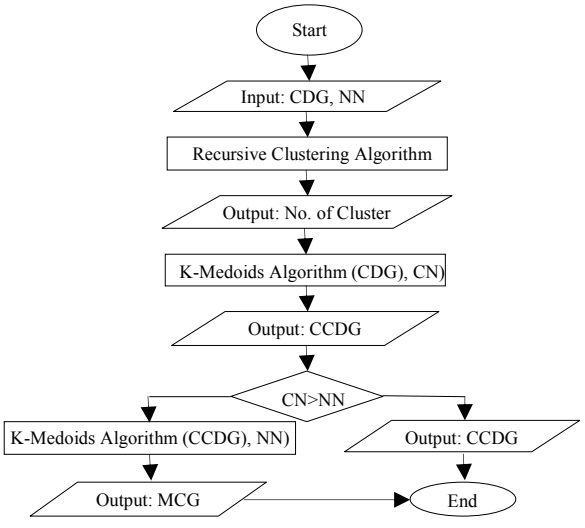


Figure 5. The steps of the double K-Medoids algorithm.

4. System Cost Model

Each cluster has both communication cost with other clusters and workload cost that specifies the total workload assigned to it. So, we have to model both costs as follows in the next subsections.

4.1. Communication Cost Between Clusters

The DOOP model provides an accurate representation for the communication activities among nodes in a DOO system. It also provides a way to evaluate the execution cost of the software modules and their related communication activities. The performance model consists of two main parts: The execution server and the communication server. The execution server and its related analysis represents and evaluates the execution cost of the software modules that reside on each node [6, 18].

The communication server provides the analysis representing the communication activities (including objects updating) of this node with other nodes and the evaluation process for communication cost. The total cost will be the summation of both execution and communication costs [38]. In the following, we are going to describe in details the model evaluation for the communication process. Assume that the overall arrival rate to the communication queue λ_{ck} which is given by [18]:

$$\lambda_{ck} = \lambda_{cs} + \lambda_{cb} + \lambda_{cu} \tag{2}$$

Where: λ_{cs}, λ_{cn} and λ_{cu} represent the communication arrival due to External User Request (EUR), Remote Request (RR) and updating objects data on other nodes, respectively:

$$\begin{aligned} \lambda_{ck} &= \beta_s \lambda_s \\ \lambda_{cn} &= \beta_n \lambda_n \\ \lambda_{cu} &= \sum_{i=1}^{N} \lambda_{Ui} \quad \lambda_{cui} = P_{i1}\lambda_s + P_{i2}\lambda_n \end{aligned} \tag{3}$$

Where, β_s and β_n are the message multipliers for EUR and RR. Let λ_{cui} be the arrival rate corresponding to $object_i$ data updating. Since, the updating process to an $object_i$ occurs due to processing EUR or RR, P_{i1} is defined to be the probability that $object_i$ is updated due to EUR and P_{i2} is the probability that $object_i$ is modified due to RR. Therefore, the expected communication service time for each class will be:

$$t_{cs} = \frac{m_s}{R}, \; t_{cn} = \frac{m_n}{R}, \; t_{ui} = \frac{m_{sui}}{R} \tag{4}$$

Where, c, t_{cs}, t_{cn} and t_{ui} are the expected communication service time for EUR, RR and for update requests from $object_i$. While m_s, m_n and m_{ui} are the expected message sizes of EUR, RR and of sending $object_i$ updating data and R is the communication channel capacity.

Furthermore, the average communication service time for node (k) will be:

$$t_{ck} = P_{cs}\, t_{cs} + P_{cn}\, t_{cn} + \sum_{i=1}^{N} P_{ui}\, t_{ui} \tag{5}$$

Where $P_{cs} = \frac{\lambda_{cs}}{\lambda_{ck}}$, $P_{cn} = \frac{\lambda_{cn}}{\lambda_{cs}}$, $P_{ui} = \frac{\lambda_{ui}}{\lambda_{ck}}$.

P_{cs}, and P_{cn} are the probabilities of activating communication service by the external user requests and by remote request, respectively. P_{ui} is the probability of sending $object_i$ data update to other nodes. If we assume that each individual class will be allocated to a separate node in the DOOP performance model, we can use Equation 3 to compute the average cost for communication between a specific class and all other classes in the system.

However, if we further conducted this evaluation on the classes in a pair-wise fashion, we can get the communication cost between each and every two classes within the object oriented distributed system.

As shown in Figure 6, in CDG each class is represented as a vertex and the communication between classes is represented by undirected weighted edges. For example, an edge between class A and B represents a communication activity that exists between these two classes due to either data transfer or classe's dependency [1]. The weight of the edge W_{AB} represents the cost of that communication activity. If no data communication or relationship dependency has been recognized in the object oriented application between two classes, no edges will connect them in the CDG [16, 33].

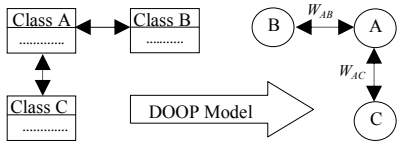


Figure 6. A Graph representation for interclass communication.

At this point, we propose to use the DOOP model to evaluate the time cost for communications that occurs between different classes in the object oriented system. The computed values are then used as the weights assigned to the corresponding edges in the CDG. The communication cost between two clusters can be computed according to the following equation:

$$T_{V,K} = \sum_{i,j=1} W_{ij} \tag{6}$$

Where, i, j the two objects connected located at different clusters, v, k, W_{ij} communication cost between objects i, j.

4.2. Cluster Workload Cost

With commercial supercomputers and homogeneous clusters of PCs, static load balancing is accomplished by assigning nearly equal tasks to each processor. With heterogeneous clusters, the system designers have the

option of quickly adding newer hardware that is more powerful than the existing hardware. This paper assumes using homogenous processors [14, 38]. The average workload cost of each cluster can be computed by summing the average workload of each object in the cluster and dividing it by the total number of objects in the cluster. Assuming that the execution time required for each object T_i is known, then we can compute the average workload cost of the cluster C_j as follows [21]:

$$T_{Ci} = \frac{\sum_{i=1}^{M} T_i}{M} \quad (7)$$

Where, T_i is the execution time of object$_i$, M is the number of objects in cluster$_j$.

The average workload for the clusters in the system can be computed as the summation of average workloads of each cluster divided by the number of clusters as follows:

$$T_{average} = \frac{\sum_{j=1}^{N} C_i}{N} \quad (8)$$

4.3. Cost Function Development

One of the main objectives of this work is to improve the overall system utilization by distributing and balancing the processor time among the execution and the communication processes based on the module's needs. Therefore, the overall time cost function equation can be written as:

$$T_{Overall\ Cost} = \gamma * T_{Workload\ Cost} + (1 - \gamma) * T_{Communication\ Cost} \quad (9)$$

Where, $T_{OverallCost}$ is the overall cost of the system, γ is a parameter expressing the balance between the two parts, whose value is constant that varies between (0 and 1) and used to distinguish the importance of each cost term.

- $T_{Communication\ Cost}$: Communication cost between clusters, which is the part of the cost-function which is minimize the inter-processor communication cost minimized.
- $T_{Workload\ Cost}$: Average workload in overall clusters, which is minimal when balance the workload of processors.

The performance of clustered architectures relies on steering schemes that try to find the best trade-off between workload balances and inter communication cost in clusters [21]. We have modified the clustering algorithms for producingbalanced workload clusters first by obtaining the clusters, which have minimum communication. Then, we calculate the total workload in each cluster. If they are not balanced, we re-cluster them. We use the Mean Square Error (MSE) for measuring the workload balance.

4.4. Using MSE to Evaluate the Cluster Performancel

To ensure the balance of the workload assigned to the clusters, we first cluster the objects minimizing the communication cost between the clusters; Then we measure the average workload of each cluster and compare it to the average workload for all objects. If the cluster's workload is equal or almost equal (i.e., less or greater than it by a certain threshold) to the average workload, then we accept this clustering.

But, when cluster's workload is far beyond the average workload, the clustering algorithm will be performed again until the balanced workload clusters is reached which gives the least overall MSE. The MSE is computed using the following Equation:

$$Overall\ MSE = \frac{\sum_{j=1}^{N} (T_{Cj} - \mu)^2}{N} \quad (10)$$

Where, T_{Cj} cluster average workload (computed by Equation 7), μ average workload ($T_{average}$) (computed by Equation 8), N number of clusters (equals to no. of processors in the system).

5. Simulation Results

We have created a simulation program using MATLAB7.10.0.499 which, given the task and the processor graphs, generates mappings and computes the total workload time in each cluster and the MSE. We have also created a random graph generator and random execution time for each object. The experiments are aimed to determine the performance of the various clustering algorithms when trying to achieve the workload balance while achieving the least possible communication produced by the clustering.

We measure the balance of these clusters by computing the MSE value. The first group of experiments investigates the ability of the various clustering algorithms to achieve the balance of the workload (in terms of MSE) when clustering different number of objects to 3 and 4 clusters. The matrix of the CDG was randomly generated with random communication values and execution times for all objects. We present the results of comparisons between the previous clustering algorithms and the three proposed algorithms in Figures 7 and 8.

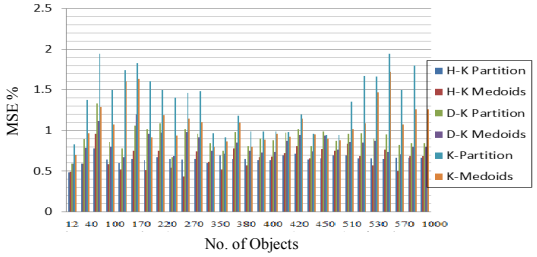


Figure 7. MSE% of the clusters workload produced by different clustering algorithms at 3 clusters.

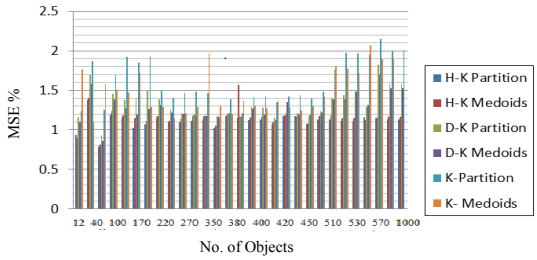


Figure 8. MSE% of the clusters workload produced by different clustering algorithms at 4 clusters.

We note from these figures that all the algorithms, except K-Partition algorithm, from 300 to 420 objects, produce almost the same workload MSE results. The K-Partition algorithm gives the highest MSE values, which means that it is the worst performance algorithm.

On the other hand, the H-K Partition and H-K Medoids algorithms, at more than 420 objects, give the least workload MSE which means that they are better than the other algorithms in achieving the balanced clusters at high number of objects. While the D-K Partition, D-K Medoids and K-Medoids algorithms are the next level in quality.

In the second group of experiments, we investigate the performance of the various clustering algorithms at variable number of clusters and fixed number of objects. The matrix of the CDG was randomly generated with random communication values and execution times for all objects. Figures 9, 10 and 11 give these results.

From these results we conclude that the K-Partition algorithm produces the highest MSE values while the H-K Partition, the H-K Medoids clustering algorithms produces the least MSE values (about 13% for the different number of clusters). In Figure 11, the results came to be consistent with the results discussed in the case study presented above.

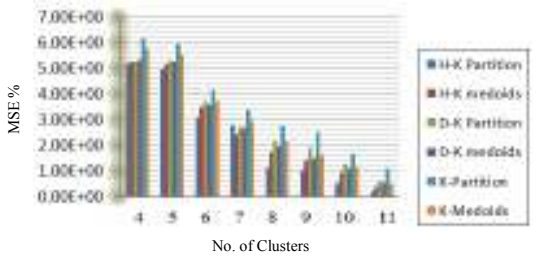


Figure 9. MSE% of the clusters workload produced by different clustering algorithms at variable number of clusters for 400 objects.

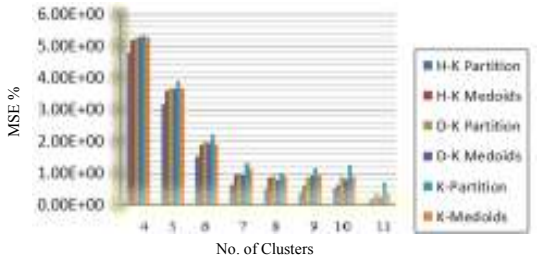


Figure 10. MSE% of the clusters workload produced by different clustering algorithms at variable number of clusters for 500 objects.

However, the H-K Medoids, D-K Medoids and D-K Partition algorithms gave results close to each other. Thus we conclude that, the H-K Partition algorithm produce better clusters quality where these clusters have both least communication between clusters and balanced workload.

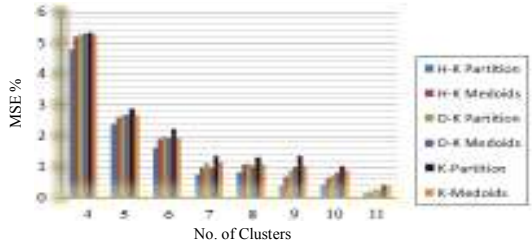
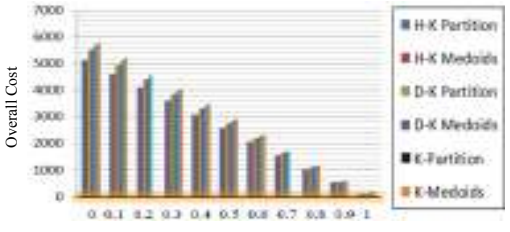
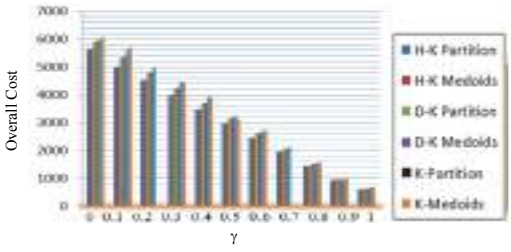


Figure 11. MSE% of the clusters workload produced by different clustering algorithms at variable number of clusters for 550 objects.

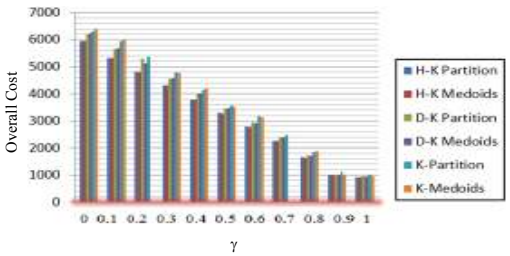
The third group of experiments has been built for evaluating the performance of the system using the overall cost function at different conditions and different values of γ using the proposed and the old clustering algorithms.



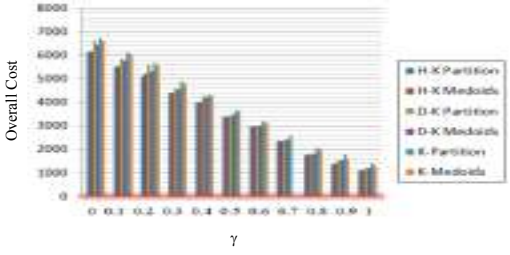
a) 150 objects (10 clusters).



b) 300 objects (10 clusters).



c) 500 objects (10 clusters).



d) 800 objects (10 clusters).

Figure 12. The Overall Cost by different clustering algorithms at different values of γ.

The fourth group of experiments has been built for evaluating the performance of the system using the overall cost function at γ=0.5 value using the proposed and old clusters algorithms at 3 and 4 clusters as shown in Figures 13 and 14.

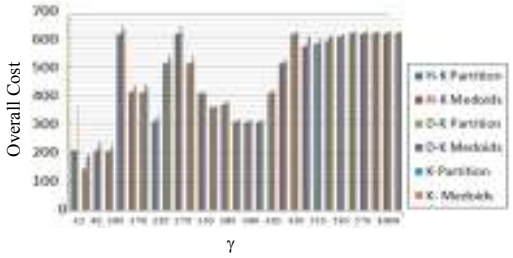


Figure 13. Overall cost function by different clustering algorithms at γ=0.5 when partitioned to 3 clusters.

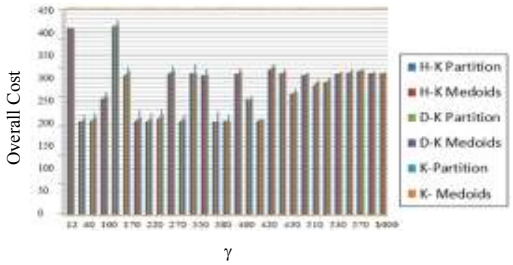


Figure 14. Overall cost function by different clustering algorithms at γ=0.5 when partitioned to 4 clusters.

From the results, comparing the proposed clustering algorithms with old clustering algorithms in the different cases, the simulation results show that a better performance can be achieved by the proposed algorithms. However, we find that the proposed algorithms (H-K Partition algorithm and H-K Medoids algorithm) give better results than the other algorithms about 5.4% at γ(0 to 0.5).

However, D-K Partition algorithm and D-K Medoids algorithm give the second best out come around 3.7% and close to each other from γ=0.6 to 1. Behind that, the K-Partition algorithm and K-Medoids algorithm give worst results and the K-Partition give the worst values. We also, found out that the balance causes minimum total cost at γ=0.5.

6. Conclusions

Load balancing is a computer networking methodology to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time and avoid overload. In this paper, we presented three different clustering algorithms that cluster the distributed objects into a given number of clusters. These algorithms aim first to minimize the communication cost between the clusters and to ensure that these clusters are balanced in terms of the workload assigned to each node.

The results showed that the hierarchical and K-Partition algorithm provides the minimum MSE in deferent cases. Attention concentrate for most important consequence is the dispersion coefficients degrade with increasing the number of clusters. For their more, the H-K Partition and H-K Medoids authenticate the normal distribution.

We proposed a cost function for enhancing the overall system utilization by distributing and balancing the processor time between the execution and the communication processes in the basis of the module's needs. Next, the experimental results showed that the performance of the proposed algorithms: H-K Partition and H-K Medoids have the best clustering quality. While in the next place comes the D-K Partition and D-K Medoids algorithms.

References

[1] Abdel-Raouf A., Fergany T., Ammar R., and Hamad S., "Performance-Based Modeling for Distributed Object-Oriented Software," *the Journal of Computational Methods in Sciences and Engineering Archive*, vol. 6, no. 5, pp. 769-773 , 2006.

[2] Abella J. and Gonzlez A., "Inherently Workload-Balanced Clustered Microarchitecture," *in Proceedings of the 19th International Parallel and Distributed Processing Symposium*, Denver, Colorado, pp. 20, 2005.

[3] Abu Abass O., "Comparisions between Data Clustering Algorithms," *the International Arab Journal of Information Technology*, vol. 5, no. 3, pp. 320-325, 2008.

[4] Appavoo J., "Clustered Objects," *A PhD Thesis*, *Department of Computer Science*, University of Toronto, 2005.

[5] Babnik T., Aggarwal R., and Moorep,. "Principal Component and Hierarchical Cluster Analyses as Applied to Transformer Partial Discharge Data With Particular Reference to Transformer Condition Monitoring," *IEEE Transactions on Power Delivery*, vol. 23, no. 4, pp. 2008-2016, 2008.

[6] Banerjee A. and Ghosh J., "Scalable Clustering Algorithms with Balancing Constraints," *Data Mining and Knowledge Discovery Journal*, vol. 13, no. 3, pp 365-395, 2006.

[7] Banerjee A., Chandola V., and Kumar V., "Anomaly Detection: A Survey," *ACM Computing Surveys Journal*, vol. 41, no. 3, pp. 1-72, 2009.

[8] Bereson A. and Lobbia R., "Efficient Track-to-Track Assignment using Cluster Analysis," *in Proceedings of the 9th IEEE International Conference on Information Fusion*, Florence, Italy, pp. 1-8, 2006.

[9] Dash M., Liu H., Scheuermann P., and Tan K., "Fast Hierarchical Clustering and its Validation," *Data and Knowledge Engineering Journal*, *Elsevier*, vol. 44, no. 1, pp. 109-138, 2003.

[10] Ding J., Ding T., and Meulen P., "Throughput Analysis of Linear Cluster Tools," *IEEE Transancations on Automatic Science Engineering*, vol. 1, no. 1, pp. 104-109, 2005.

[11] Elsősser R., LŰcking T., and Monien B., "New Spectral Bounds on k-Partitioning of Graphs," *in Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, New York, USA, pp. 255-262, 2001.

[12] Garama A., Gupta A., Karypis G., and Kumar V., *Introduction to Parallel Computing*, Addison-Wesley, 2003.

[13] Gibson T., "How Things Work: Standard Deviation as a Tool for Measuring Precision and Accuracy," available at: http://archives. profsurv.com/magazine/article.aspx?i=1826, last visited 2007.

[14] Grosu D. and Chronopoulos A., "Algorithmic Mechanism Design for Load Balancing in Distrributed Systems," *IEEE Transactions on Systems*, *Man and Cybernetics-Part B: Cybernetics*, vol. 34, no. 1, pp. 77-84, 2004.

[15] Guadalupe J., Basnet B., Andrew H., Sung S., and Bernardete M., "Fuzzy Clustering and Data Analysis Toolbox for using with Matlab," available at: ftp://ftp.unicauca.edu.co/cuentas /fiet/docs/DEIC/Materias/computacion%20intelig ente/parte%20II/semana12/clustering/mfiles/Clus teringToolbox/FuzzyClusteringToolbox.pdf, last visited 2008.

[16] Hamad S., Fergany T., Ammar R., and Solit S., "Mapping Distributed Object-Oriented Software to Architecture with Limited Number of Processors," *in Proceedings of IEEE International Symposium on Signal Processing and Information Technology*, Giza, Egypt, pp. 531-536, 2007.

[17] Hamad S., Fergany T., Ammar R., and Abd El-Raouf A., "A Double K-Clustering Approach for Restructuring Object-Oriented Software," *in Proceedings of IEEE Symposium on Computers and Communications*, Marrakech, pp. 169-174, 2008.

[18] Hamad S., Khalifa M., Ammar R., and Soleit E., "Performance-Based Restructuring of Distributed Object-Oriented Computations for a Cluster of Multiprocessors," *A PhD Thesis*, Ain Shams University, 2008.

[19] Hongjin J., Zeng D., Yanxiang S., Yangang W., and Xisheng W., "Semi-Hierarchical Correspondence Cluster Analysis and Regional Geochemical Pattern Recognition," *the Journal of Geochemical Exploration*, vol. 93, no. 2, pp. 109-119, 2007.

[20] Hősel V. and Walcher S., "Clustering Techniques: A Brief Survey," *Technical Report*, Insttitut fiir Biomathematik Und Biometrie GSF, Research Center, Germany, 2000.

[21] Huang C., Zhou G., Abdelzaher T., Son S., and Stankovic A., "Load Balancing in Bounded-Latency Content Distribution," *in Proceedings of the 26th IEEE International Symposium on Real-Time Systems*, Miami, USA, pp. 61-73, 2005

[22] Jiang D., Tang C., and Zhang A., "Cluster Analysis for Gene Expression Data: A Survey," *IEEE Transactions on Knowledge and Engineering*, vol. 16, no. 11, pp. 1370-1386, 2004.

[23] Kim T. and Shin Y., "Role-based Decomposition for Improving Concurrency in Distributed Object-Oriented Software Development Environments," *in Proceedings of the 23rd IEEE Conference on Computer Software and Applications*, Arizona, USA, pp. 410-415, 1999.

[24] Kotsiantis S. and Pintelas P., "Recent Advances in Clustering: A Brief Survey," *WSEAS. Transactions on Information Science and Applications*, vol. 1, no. 1, pp. 73-81, 2004.

[25] Lee Y. and Antonsson E., "Dynamic Partitional Clustering using Evolution Strategies," *in Proceedings of the 26th IEEE Annual Conference on Industrial Electronics Society*, Nagoya, Japan, pp. 2716-2721, 2000.

[26] Lee Y. and Song Y., "Selecting the Key Research Areas in Nano-Technology Field using Technology Cluster Analysis: A Case Study Based on National," *Technovation*, vol. 27, no. 1, pp. 57-64, 2007.

[27] Macnab J., Miller L., and Polatajko H., "The Search for Subtypes of DCD: Is Cluster Analysis The Answer?," *Human Movement Science*, vol. 20, no. 1, pp. 49-72, 2001.

[28] Narayanan S., Ozturk O., Kandemir M., and Karakoy M., "Workload Clustering for Increasing EnergySavings on Embedded MPSoCs," *in Proceedings of the IEEE International Conference SOC*, VA, USA, pp. 155-160, 2005.

[29] Roy S. and Chaudhary V., "Strings: A High-Performance Distributed Shared Memory for Symmetrical Multiprocessor Clusters," *in Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*, Chicago, USA, pp. 90-97, 1998.

[30] Smith M. and Munro M., "Runtime Visualisation of Object Oriented Software," *in Proceedings of the 1st IEEE International Workshop on Visualizing Software for Understanding and Analysis*, Washington, USA, pp. 81-89, 2002.

[31] Sneed H. and Dombovari T., "Comprehending a Complex, Distributed, Object-Oriented Software System A Report from the Field," *in Proceedings of the 7th International Workshop on Program Comprehension*, PA, USA, pp. 218-225, 1999.

[32] Thain D., "Coordinating Access to Computation and Data in Distributed Systems," available at: http://research.cs.wisc.edu/htcondor/doc/thain-dissertation.pdf, last visited 2004.

[33] Todd C., Toth T., and Busa-Fekete R., "GraphClus, a MATLAB Program for Cluster Analysis using Graph Theory," *Computers and Geosciences*, vol. 35, no. 6, pp. 1205-1213, 2009.

[34] Tsai S., Chiou J., and Jen H., "Load Balance Facility in Distributed MINIX System," *in Proceedings of the 20th System Architecture and Integration Conference*, Liverpool, UK, pp. 162-169, 2002.

[35] Wang X., Yan Z., and Xue W., "An Adaptive Clustering Algorithm with High Performance Computing Application to Power System Transient Stability Simulation," *in Proceedings of the 3rd International Conference on Electric Utility Deregulation and Restructuring and Power Technologies*, Nanjuing, China, pp. 1137-1140, 2008.

[36] Wu X., Taylor V., and Sharkawi S., "Performance Analysis and Optimization of Parallel Scientific Applications on CMP Cluster Systems," *in Proceedings of International Conference on Parallel Processing*, Washington, USA, pp. 188-195, 2008.

[37] Xing Z. and Stroulia E., "Understanding Class Evolution in Object-Oriented Software," *in Proceedings of the 12th IEEE International Workshop on Program Comprehension*, Washington, USA, pp. 34-43, 2004.

[38] Zhang Q., Riska A., Sun W., and Smimi E., Ciardo G., "Workload-Aware Load Balancing for Clustered Web Servers," *Parallel and Distributed Systems*, *IEEE Transactions on Computer Society*, vol. 16, no. 3, pp. 219-233, 2005.

Heba Ragab received the BSc degree in 2000 and MSc in 2007, and PhD degree in 2014, in computer and automatic control from the Faculty of Engineering, Tanta University. She is working now as a Lecturer at Computers and Automatic Control Department., Tanta University., Egypt. Her interests are in the area of: Distributed systems and computations, software restructuring and neural networks.

Amany Sarhan received the BSc degree in electronics engineering and MSc degree in computer science and automatic control from the Faculty of Engineering, Mansoura University, in 1990 and 1997, respectively. She awarded the PhD degree as a joint research between Tanta Universety, Egypt and University of Connecticut, USA. She is working now as an Associate Prof. at Computers and Automatic Control Department., Tanta University, Egypt. Her interests are in the area of: Distributed systems and computations, software restructuring, schema matching, image and video processing.

Al Sayed Sallam received his MSc and PhD degrees from Bremen in Germany on 1983 and 1987 respectively. He is working now as an Associate Prof. and Head of Computers and Automatic Control Department., Tanta University, Egypt. His interests are in the area of: Control, software restructuring, robotics and network. He is the CIO of Tanta University for the last 2 years.

Reda Ammar received his PhD degree, University of Cyhhonnecticut, computer science, 1983. He worked as the head of Department at Computer Science and Engineering Department, UCONN, USA. His Research Interests are: Software performance engineering; parallel and distributed computing; real-time systems and cluster and grid computing. He is IEEE (senior member), ACM, ISCA, Editor-in-Chief of the International Journal of Computers and Their Applications, Associate Editor in Computing Letters and Member of the Board of Directors of the International Society of Computers and Their Applications.