Query Driven Data Subspace Mapping

Panagiotis Fountas[1 [0000−0002−4555−6606]], Maria Papathanasaki[1 [0000−0002−7440−8485]], Kostas Kolomvatsos[1 [0000−0002−9442−3340]], and Christos Anagnostopoulos[2 [0000−0003−1517−6757]]

[1] Department of Computer Science and Telecommunications, University of Thessaly, Papasiopoulou 2-4, 35131, Lamia, Greece
[2] School of Computing Science, University of Glasgow, Lilybank Gardens 18, G12 8RZ, Glasgow, UK
{pfountas,mpapathanasaki,kostasks}@uth.gr, christos.anagnostopoulos@glasgow.ac.uk

Abstract. The increased use of multiple types of computer systems and smart devices in several areas, has created massive amounts of data. Concurrently, the need for a subset of these data by numerous applications and users for task execution and knowledge extraction, has resulted in the injection of a massive number of queries per second into distributed database servers. As a result of this phenomena, a major process is the efficient response of these queries both in terms of time and the detection of acceptable data, while rejecting undesired data points. In this paper, we present a hierarchical query-driven clustering approach, for performing efficient data mapping in remote databases for future incoming queries. We distinguish ourselves from current methods, by combining the technique of Query-Based Learning (QBL) with a hierarchical clustering of multiple forms of clustering in the same model. The suggested model's performance is assessed, using a number of experimental scenarios as well as numerical data.

Keywords: Data mapping · Data Management · Query-Based Learning · Hierarchical Clustering · Data Retrieval.

1 Introduction

In the era of the intense escalation of the data production, analysts must cope with a massive amount of data which are spread across multiple databases. As a consequence, several challenges have emerged for data management and information extraction, such as data caching, data quality assurance through the elimination of anomalies, missing values and data mapping. Data mapping is the process of combining data from various datasets into a particular dataset and storing it in a consistent manner. This procedure is essential in various operations, including data migration, data integration, and data warehousing.

In this paper, we present a hierarchical clustering model for detecting the suitable data for executing queries delivered to a server by users in the shortest time possible. We present a model that takes into consideration the required

data of a number of queries that arrive in the server over a period of time, allowing us to detect and collect the data that a future query will request, without requiring to scan the distributed databases. We focus on the usage of two distinct methods of clustering algorithms, i.e. fuzzy clustering and hard clustering, which are implemented using the Fuzzy C-Means (FCM) and K-Means (KM), respectively. These clustering methods are used to classify comparable queries into clusters, based on the data required for execution. The suggested model examines the queries that come into the server and identifies the clusters that are the most related to them. Based on previous research, we include a mechanism in our model that calculates the overlap of the area of interest between two queries. The novelty of this paper is that we combine the QBL [13][14] with a hierarchical clustering scheme into a model that is able to predict which of the data it has to retrieve for similar future queries. The main contributions of this paper are as follows:

- We propose an hierarchical clustering-based model combining two different types of clustering for the detection of the appropriate data in the minimum possible time and with the minimum error.
- We adopt QBL to retrieve the appropriate data for the user's query, relying on the retrieved data of previously executed queries.
- We argue on an area overlap metric between the areas of the queries to detect the existence of data points of common interest.

2 Related work

The data management processes constitute an important factor for the effectiveness of various tasks and operations, such as the extraction of analytics from a set of data etc. The research community has proposed several models and mechanisms for the improvement of various data management processes. Caching is a crucial data management process which prevents the burden of the memory with unnecessary data. The authors of [1] present a solution for the assigning of the queries and tasks in the appropriate edge computing nodes in order to reduce as much as possible the response time. For this purpose, the authors propose a method for the estimation of the computational burden, that an allocation of a query will be added to a node. Also, the authors develop an ensemble similarity scheme, responsible to deliver the complexity class for each query or task and a probabilistic decision-making model. In [2] the authors define the concept of a Query Controller (QC) that assigns each of the queries into a processor, which is placed in front of each data partition. Based on this technology, they develop a framework for query assignment which involves two learning schemes, i.e., a Reinforcement Learning (RL) and a clustering scheme. Also, they propose a multiple Q-tables scheme as knowledge base of the QC in the RL case and a technique for deriving the level of compactness of the created clusters in the clustering scheme, to deliver the best possible QP for each assignment. In [3] the authors introduce an adaptive, reciprocity-based Machine Learning mechanism, to estimate the answers of a variety of aggregate queries (AQs) avoiding the big

data back-end. The mechanism learns from past analytical-query patterns while they develop solutions to correspond in the changes in queries' analytics and analysts' interests. In [4] Data Canopy framework is introduced to reduce the time needed to compute statistics. It is a smart cache designed for statistical exploratory analysis. Data Canopy calculates and caches the fundamental primitives of statistical measures, allowing it to compose results for future queries without having to return to the base data. More specifically, it deconstructs statistics into their underlying primitives, caches and retains them, while using them to generate future computations of statistics either on the same or on overlapping data. The authors of [5] propose WANalytics, a geo-didtributed system that copes with the Wide-Area Big Data problem, a challenge that concerns the supporting rich Directed Acyclic Araphs of computation, over globally distributed data. WANalytics is formed by two major parts. The first one is a runtime layer that distributes user DAGs around data centers, and the other part pertains to a workload analyzer that constantly checks and improves the user workload. While maintaining data domination requirements, WANalytics transfers computation to edge data centers, improves workflow efficiency, and duplicates the base data when needed.

3 Preliminaries

3.1 K-Means

KM is one of the most popular and widely used unsupervised clustering algorithms. It groups the given data into K clusters trying to minimize the following objective function:

$$J = \sum_{i=1}^{K} \sum_{p \in G_i} \|p - g_i\|^2 \tag{1}$$

The minimization of equation (1) is equivalent to the minimization of distance of points in a cluster G_i with the centroid g_i. We consider a set $D = \{\overrightarrow{p}_1, \overrightarrow{p}_2, \ldots, \overrightarrow{p}_n\}$ which consists of n vectors each one can be considered as a d-dimensional point. KM has as goal to split the n vectors into K clusters, where K $\leq$ n and must be defined in advance. The clusters have to have been created in such way such that $G = G_1 \cup G_i \cup , \ldots, \cup G_K$ where $G_i \subset G, \forall i \in [1, K]$ and $G_i \cap G_j = \varnothing, \forall i, j \in [1, K]$. Each cluster is represented by its centroid. The steps of the KM algorithm are described below [8][9][10]:

- At first, K points from D are randomly selected to be the centroids of the clusters.
- In the second step, the algorithm computes for every point in D the distance from every centroid. Hence, the algorithm assigned the examined point to the cluster with nearest centroid. The Euclidean distance is the most widely adopted metric for this step.
- Afterwards the algorithm recalculates the centroids of each cluster. The new value for each one of d dimensions is equal to the average value that the members have in that dimension.

– The algorithm returns to the second step in case the criterion function does not become the minimum, i.e., the clusters does not remain consistent.

We have to note that the centroids of the clusters may not contained in the dataset D, since it will arise from the previous iteration process. The time complexity of KM depending on three parameters; the number n of data vectors in D, the number of clusters and the number of iterations l that the algorithm needed to cluster the data. Consequently, the time complexity is $O(n \cdot K \cdot l)$ [6]

3.2 Fuzzy C-Means

FCM algorithm is a fuzzy clustering algorithm which assigns each data point into a cluster according to membership value $u_{ik} \in [0, 1]$. The membership value indicates the correlation between a data point and the center c_i of the respective cluster and is assigned to the cluster with the highest membership value [7]. Given a dataset D of n vectors each one has d dimensions the FCM outputs a $n \times \Gamma$ matrix $\mathbf{U}$ which includes the membership values for each data point, and a set $C = \{C_1, C_2, C_3, \ldots, C_\Gamma\}$ that contains the created clusters. The FCM requires to be pre-defined three parameters: the fuzzier $m \in [1, \infty)$ that controls the fuzziness of the clustering, the number of clusters Γ and the stopping criterion value $\beta \in [0, 1]$ and intend to minimize the following objective function:

$$J_m(\mathbf{U}, C) = \sum_{i=1}^{\Gamma} \sum_{k=1}^{n} (u_{ik})^m \cdot \|p_k - c_i\|^2 \tag{2}$$

The steps of the algorithm are referred below [8][9][11][12]:

– Choosing of the parameters m,Γ,β
– Initializing the membership matrix $\mathbf{U}$
– Calculating the centers of every cluster in C
– Updating the membership matrix $\mathbf{U}$
– Repeating steps 3,4 until the divergence is less than β
– Output $\mathbf{U}$,C

The updating of the u_{ik} and the c_i for each cluster are calculated by the equations (3),(4) respectively.

$$u_{ik} = \frac{1}{\sum_{j=1}^{\Gamma} \left\{ \frac{\|p_k - c_i\|}{\|p_k - c_j\|} \right\}^{\frac{2}{m-1}}} \tag{3}$$

$$c_i = \frac{\sum_{k=1}^{n} (u_{ik})^m \cdot p_n}{\sum_{k=1}^{n} (u_{ik})^m} \tag{4}$$

4 Problem Description

In our scenario we consider a set $DB = \{DB_1, DB_2, \ldots, DB_w\}$ of geo-distributed databases(DDBs) and a server (SV). Also, we suppose that a group Z_i of Internet of Things (IoT) devices is connected with a DDB DB_i such that each Z_i is connected with only one DB_i and vice versa. The IoT devices collect and report data into the respective DB_i with form of multivariate vectors i.e., $X_j^t = [x_1^t, x_2^t, \ldots, x_d^t]$, where the index j expresses the IoT device that reported the vector and the index t shows the time instance that the vector was reported. The DDBs receive the multivariate vectors and store them in appropriate format to be ready for further processing activities. The SV receives queries $Q = \{q_1, q_2, \ldots, q_z\}$ from applications and users. We consider that every vector can be represented as a point in a d-dimensional space. Every query q_i requires a number Φ of points as answer. Every query q_i can contain range selection operators for one or more dimensions of the data, creating the boundaries of the area in which the data are located. The SV tries to detect the appropriate data as answer to the incoming queries in the minimum possible time. In this paper we propose a mechanism for the mapping of the data that the queries need. Our mechanism is 'triggered' every time a query is sent by an application or user. More specifically, we focus on a hierarchical clustering process where we try to group the incoming query with previous received queries. The proposed model performs two types of clustering: (a) a fuzzy clustering where the incoming queries are assigned into one or more clusters; (b) a hard clustering to identify the subspace where the required data are located.

5 Adopted Methods in Data Mapping Process

The first of the methods for the data mapping is the baseline method (BM). This method is executed every time that a query is coming to the SV and scans the data sequentially, selecting those that satisfy the query. This approach, in particular, identifies all of the required points for every incoming query and is the optimal solution in terms of error. However, scanning all the data in the DDBs for every query, is time consuming.

The second method is called Hard Clustering Based Method (HCBM) and is based on the clustering of the 'similar' queries, that the SV receives during a period of W time instances to serve the upcoming queries. Initially, the HCBM is trained over z incoming queries, using BM to identify the data required for their execution, then utilizes KM to cluster them into groups with similar requirements. After the completion of the training phase, the HCBM method is ready to serve every incoming query in the SV. When a query is sent to the server, HCBM is 'triggered,' and it uses a correlation/similarity measurement to find the top-k similar clusters to the incoming query. In our experiments we apply the Euclidean distance metric. The HCBM uses the top-k clusters that have been detected, to retrieve data points that satisfy queries that belong in these top-k clusters.

Our proposed model is named Hierarchical Mixed Clustering Model (HMCM). The suggested model comprises two phases: the 'warming' and the 'performance' phase. The clusters and subclusters that have been built will be used in the first phase to map the appropriate data for the response to incoming queries in SV. During the warm-up stage, when a query is sent by the user to the SV, the HMCM model performs a sequential scan of the whole database to determine the suitable data points for query execution. HMCM executes hierarchical clustering after detecting the appropriate data points for each query. Initially, it uses the FCM to create a set of clusters $B = \{B_1, B_2, \ldots, B_\Gamma\}$ of the queries that were sent to the SV in previous W times. Afterward, HMCM divides each B_i cluster further into a set of $S = \{S_1, S_2, \ldots, S_K\}$ using the KM algorithm. In the second phase HMCM is activated every time a user sends a query to the SV. Firstly, the proposed model uses a similarity metric, in this case the Euclidean distance, to identify the top-k clusters whose members have the same requirements as the incoming query. The top-ℓ subspaces with which the incoming query has the most similarity, are then detected using HMCM. The adoption of the approach of overlapping between the representations of queries is a modification to produce more accurate results in the mapping of appropriate data. Each query provided by the user, is represented by the area of points that it requires as an answer. The following metric is proposed for the purpose of calculating overlap.

$$\text{Area Overlap Metric}(q_{inc}, q_{member}) = \frac{q_{inc} \cup q_{member}}{\textit{incoming query area}} \tag{5}$$

In the Area Overlap Metric (AOM) the numerator is the overlap area between two queries. The q_{inc} is the incoming query in the SV, while the q_{member} is the query which is member of one of the top-ℓ subspaces of one cluster which belongs to top-k clusters. The denominator is the area which contains the required data points for the incoming query. The result of the AOM indicates the percentage of the q_{inc} area which is covered by the area of the q_{member}. Hence, the HMCM after the detection of the appropriate clusters and subspaces, it examines the members of the detected subspaces, to find the queries with which the AOM overcomes a threshold θ and it retrieves only the data points that belong to them. The approach of selection of the members gives the ability to obsolete queries that belong into the top-ℓ subspaces of top-k clusters, but they do not require data from a common area. This approach also allows the HMCM to ignore queries with which the incoming query shares a common area, but the overlap is less than the threshold, preventing the HMCM from increasing the error in its predictions.

6 Experimental Setup and Evaluation

The experimental evaluation of the proposed model is relying on the Query Analytics Workloads Dataset The dataset contains three files of range/radius query workloads from Gaussian distributions over a real dataset. In our experiments we focus on range queries, and we are based on the file Range Queries Aggregates to

create three datasets which are named Warming Dataset (D_W), Dataset of two-dimensional points (D_{2d}) and Test dataset (D_T). Each range-query in the file is stored on the following format $\{X, Y, Xr, Yr, Count, SUM, AVG\}$. However, we take into consideration only the first four attributes which refer to the range query. The first two attributes are the coordinates for the x-axis and y-axis for the center of rectangle respectively. The third and fourth attribute represent the ranges of the first two attributes. The D_w consists of 1.000 queries of the format $q_i = \{X_i, Y_i, Xr_i, Yr_i\}$. We randomly generate for each q_i a number $\omega_i \in [20, 30]$ of two-dimensional data points which are located inside the rectangle of q_i. The total number of two-dimensional points is equal to 24.923 and constitutes the D_{2d} dataset. The last dataset D_T contains $\psi = 1.000$ incoming Range queries with the same distribution and format with the D_w. Our goal is to confirm that the proposed model has the ability to detect the data that an incoming query into an SV needs. We use the D_w to 'train' the HCBM and HMCM model while the D_T dataset is used to test the performance of the models BM, HCBM and HMCM.

We evaluate the described models both the error levels and the time that they need to detect the data. The evaluation of the models is relying on the metrics of Precision (PRE), Recall (REC), Accuracy (ACC), False Positive Rate (FPR) and F1-score (FSC) as they ensue from the calculation of True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN). We define as TP the number of data that an incoming query q_i needs and they detected while TN the number of data that the q_i does not need and the model correctly reject them. On the other hand, FP is the number of data that the q_i does not need but the model retrieved them, and FN is the number of data that the q_i needs but the model does not detect them. The required time for each query is symbolized as τ_i .The formulas of Precision, Recall, FPR and F1-score is defined as follows: PRE = TP / (TP+FP); REC = TP / (TP+FN) ; ACC = (TP+TN) / (TP+TN+FP+FN); FPR = FP / (FP+TN); FSC = (2 · TP) / (2 · TP+FP+FN).

The aforementioned metrics are used to calculate the performance of models for each q_i. However, the performance of the models has to be estimated over all the incoming queries, thus we use the average values of the previous metrics to calculate the overall performance of the models. Except from the metrics for the error calculation we also calculate the average time that every model needs to detect the data which satisfy the incoming queries. The overall metrics are defined as follows:

$$\mu_{\Omega_i} = \frac{\sum_{q_i \in D_T} \Omega_i}{\psi}, \Omega \in \{PRE, REC, ACC, FPR, FSC\} \tag{6}$$

The models have the best performance when the metrics μ_{Acc}, μ_{PRE}, μ_{REC}, μ_{FSC} reach the unity and the metrics μ_{FPR} and μ_{Time} are closer to zero. In our experiments we pay great attention in the metrics μ_{FPR} and μ_{Time} because the former one gives the average rate of data that the models retrieve but the incoming queries do not need them, while the latter one gives the average time

that one of the methods needs to respond into the incoming queries. We present the performance of models for different number of clusters regardless of the way that they are created i.e., from the hierarchical clustering (HMCM) or the non-hierarchical clustering (HCBM). We have to mention that in the plots for the error metrics we do not include the BM model because it detects all the requirement data without error since it scans sequentially the databases.

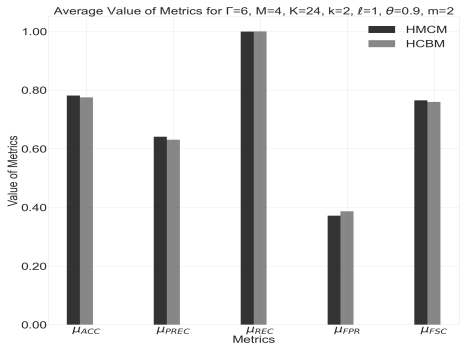

Fig. 1. Comparison between HMCM and HCBM for Γ=6, M=4 and K=24

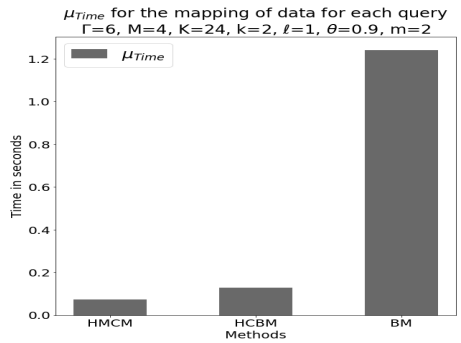

Fig. 2. Average Time comparison between the models for Γ=6, M=4, K=24

In figure 1 we compare the HMCM and HCBM when they created 24 clusters. As we can easily see, the HMCM has better performance for all metrics except the μ_{REC} where two models have the same performance. The dominance of the HMCM is revealed clearly from the figure 2 where the average time that the HMCM needs to respond to the incoming queries is significant less from the BM method and much less from the HCBM model.

Figure 3 shows the comparison of the HMCM and HCBM for 42 clusters. We observe that the HMCM overcomes the HCBM for all metrics except from the μ_{REC}, where the HMCM has slightly lower performance. Nevertheless, both μ_{FSC} and μ_{FPR} metrics confirm that the HMCM has better performance in error metrics. The figure 4 strengthens the conclusion that we deduce from figure 4 since HMCM achieve better performance in less time.

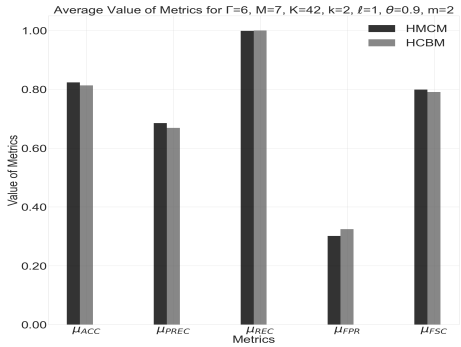

Fig. 3. Comparison between HMCM and HCBM for Γ=6, M=7 and K=42

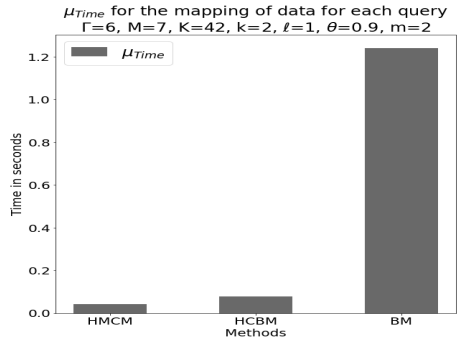

Fig. 4. Average Time comparison between the models for Γ=6, M=7, K=42

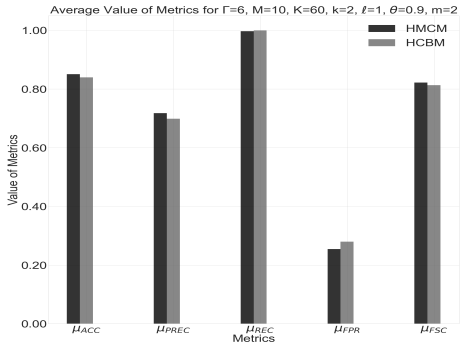

Fig. 5. Comparison between HMCM and HCBM for Γ=6, M=10 and K=60

In figures 5 & 6 the comparison of error and time metrics is presented between the models for 60 clusters respectively.In figure 5, same as previous the HMCM has better performance in the majority of the error metrics. As far as the time metric is concerned, the HMCM maps the data in less time that the other models, as it is presented in figure 6. Again, in the most important metrics

for inferring the conclusion of the designation of the best model, the HMCM clearly outperforms the HCBM.

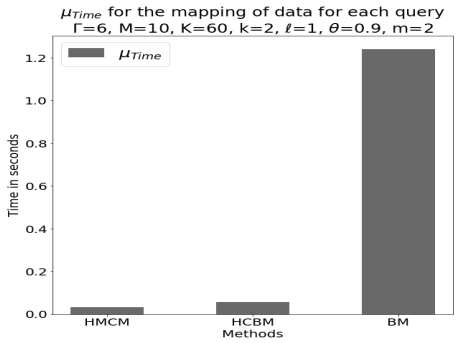

Fig. 6. Average Time comparison between the models for Γ=6, M=10, K=60

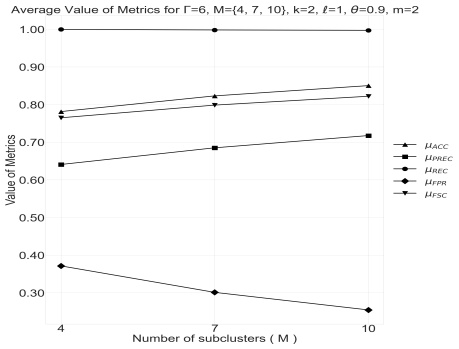

Fig. 7. Performance of error metrics for different number of subclusters in HMCM

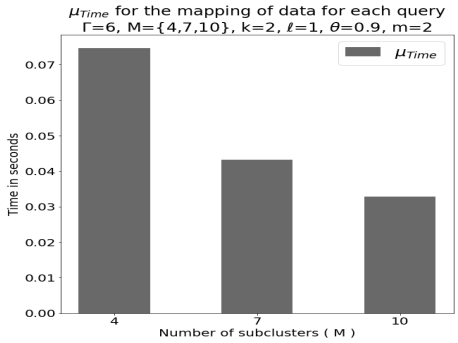

Fig. 8. Comparison of the time metric for different number of subclusters in HMCM

In figures 7 & 8 we evaluate the effect of the number of subclusters in the performance of our model both for error and time. We can easily observe that the increase of the number of subclusters clearly improves the performance of the HMCM model and simultaneously decrease the average required time for the mapping of data.

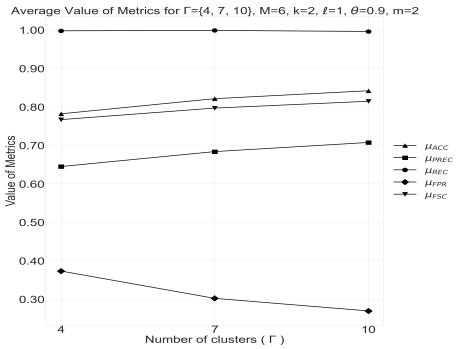

Fig. 9. Performance metrics for different number of clusters in HMCM

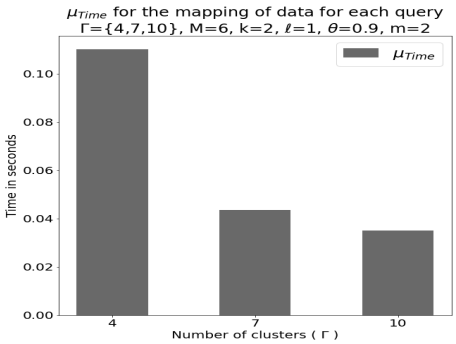

Fig. 10. Comparison of the time metric for different number of clusters in HMCM

7 Conlusions and Future Work

Data mapping is a significant data management process which plays important role in many applications domains. This process become more complex when the data are geo-distributed in different databases around the world. Data mapping can be improved if we identify relations between the data in the DDBs and the queries that the users send to the server. In this paper, we focus on the efficient mapping of data and propose a solution for an effective data mapping in the minimum possible time. We are based on the answers of past queries, and we propose an hierarchical clustering scheme which groups the past queries relying on the similarity of the requirements of the queries. Also, we involve a mechanism for the calculation of common interest data area between two queries.

We adopt this type of strategy to create small groups of queries with similar data requirements and try to benefit from the exclusion of non-similar groups with an incoming query to reduce the time of response and to prevent the caching of unneeded data. We perform an extensive set of experiments to evaluate the proposed model in order to confirm the ability of the proposed model to map the data in short time with a small amount of extra unneeded data. A possible extension of this work could be the involvement of a more complex methodology for the improvement both of error and time metrics. Also, we could adopt a deep learning model that will be able to adapt our model to changes in user requirements expressed through queries.

References

1. Kolomvatsos, K., Anagnostopoulos, C.: A probabilistic model for assigning queries at the edge. Computing. 102, 865-892, (2020).
2. Kolomvatsos, K., Anagnostopoulos, C.: Reinforcement Learning for Predictive Analytics in Smart Cities. Informatics, (2017).
3. Savva, F., Anagnostopoulos, C., Triantafillou, P.: Adaptive learning of aggregate analytics under dynamic workloads. Future Generation Computer Systems. 109, 317-330, (2020).
4. Wasay, A., Wei, X., Dayan, N., Idreos, S.: Data Canopy. Proceedings of the 2017 ACM International Conference on Management of Data. (2017).
5. Vulimiri, A., Curino, C., Godfrey, P., Jungblut, T., Karanasos, K., Padhye, J., Varghese, G.: WANalytics. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. (2015).
6. Na, S., Xumin, L., Yong, G.: Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm. 2010 Third International Symposium on Intelligent Information Technology and Security Informatics. (2010).
7. Schwämmle, V., Jensen, O.: A simple and fast method to determine the parameters for fuzzy c–means cluster analysis. Bioinformatics. 26, 2841-2848 (2010).
8. Etehadtavakol, M., Sadri, S., Ng, E.: Application of K- and Fuzzy c-Means for Color Segmentation of Thermal Infrared Breast Images. Journal of Medical Systems. 34, 35-42 (2008).
9. A. M. Jamel, A., Akay, B.: A Survey and Systematic Categorization of Parallel K-Means and Fuzzy-C-Means Algorithms. Computer Systems Science and Engineering. 34, 259-281 (2019).
10. Na, S., Xumin, L., Yong, G.: Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm. 2010 Third International Symposium on Intelligent Information Technology and Security Informatics. (2010).
11. Gupta, R., Muttoo, S., Pal, S.: Fuzzy C-Means Clustering and Particle Swarm Optimization based scheme for Common Service Center location allocation. Applied Intelligence. 47, 624-643 (2017).
12. Stetco, A., Zeng, X., Keane, J.: Fuzzy C-means++: Fuzzy C-means with effective seeding initialization. Expert Systems with Applications. 42, 7541-7548 (2015).
13. Chang, R., Hsu, H., Lin, S., Chang, C., Ho, J.: Query-Based Learning for Dynamic Particle Swarm Optimization. IEEE Access. 5, 7648-7658 (2017).
14. Albishre, K., Li, Y., Xu, Y., Huang, W.: Query-based unsupervised learning for improving social media search. World Wide Web. 23, 1791-1809 (2019).