STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIV**, Number 2, 2009

HIERARCHICAL CLUSTERING IN LARGE OBJECT DATASETS - A STUDY ON COMPLEXITY, QUALITY AND SCALABILITY

ADRIAN SERGIU DARABANT AND ANCA GOG

Abstract. Object database fragmentation (horizontal fragmentation) deals with splitting the extension of classes into subsets according to some criteria. The resulting fragments are then used either in distributed database processing or in parallel data processing in order to spread the computation power over multiple nodes or to increase data locality features on each node. In this paper we propose an analysis on the application of hierarchical clustering over object datasets (databases). We use a hierarchical clustering algorithm in order to split the object set into fragments and we analyze their quality based on data accesses in a distributed system. In order to measure the scalability of the algorithm we apply it consecutively to a small, medium and large sized database. We also compare the obtained results with those obtained with other fragmentation algorithms.

1. introduction

Splitting an object database into multiple subsets - usually named fragments - is a process used whenever one needs to distribute computations over a sets of nodes, each containing a subset of the total object database. Fragmentation is used to improve locality features of datasets, processing parallelism or a combination of both. The first approach is used when the entire object set models informations from a real world macro-entity (like an organization) that has a space distributed architecture (an organization with multiple offices located in different geographical areas). The objects are divided into

Received by the editors: November 13, 2009.

2010 *Mathematics Subject Classification.* 68M14, 68P20.

1998 *CR Categories and Descriptors.* C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks – *Distributed Systems*; E.1.2 [**Data**]: Data Structures – *Distributed Data Structures*; H.2.4 [**Information Systems**]: Database Management – *Systems*.

Key words and phrases. clustering, distributed databases.

This work is supported by the Romanian Ministry of Education in the frame of PN2 ID550/2007.

subsets according to the affinity between the real entity modeled by each object and the sets of locations. The main advantage of this approach is that each location (node) already stores the maximum amount of local pertinent information reducing thus the inter-node information exchange requirements. Most applications running in a node will handle local informations in most of the cases.

When parallel processing comes into discussion, one would like that whenever a data intensive request arrives in the (distributed) system it could be divided into multiple sub-requests, each assigned to a different processing node such that the maximum throughput is achieved. In this case data exchange minimization in not the main goal. The main goal is to parallelize the processing by dividing it over as many nodes of the system as possible. A single node is subject to become a processing bottleneck as processing is always divided over the entire system.

When both policies are needed (locality and parallelism) a request is divided in multiple tasks that are assigned to the nodes of the system such that only nodes that can provide data from their local storage to the final result participate in task resolutions.

Experience shows that distributed databases do not evolve at the pace of centralized systems. This is not due to the fact the we lack naturally distributed applications. On the contrary, most of today applications, from ticket reservation to medical patient records management, are inherently distributed. A common sense reason for this lack of development could be the complexity involved in the design and management of distributed architectures. Compared to centralized data stored, most known database distribution techniques require a lot more apriori information about the data that would be stored and applications that will run in the system than in centralized data stores. The existing algorithms for relational database fragmentation [11] are either too complicated to understand and follow or too sensitive to small changes in the inputs, making them difficult to approach by the usual database administrator (DBA).

2. Contributions

Our approach to object dataset fragmentation is based on the idea that a distributed database as described above should be easy enough to design and maintain. In this paper we present a fragmentation method using hierarchical clustering algorithm [1]. The fragmentation quality is evaluated using a partition evaluation function already used in previous work [6, 4, 2, 5, 3]. The main contribution here is to asses the validity of this partitioning method for different sized databases. We also compare our results with those obtained in other

similar works [9, 10, 4]. An application of a variant of the k-means algorithm for clustering in fragmentation is presented in [7]. Most of the other methods are based on variants of the relational database fragmentation algorithms.

We also strive to prove that the requirements of our method are far more easier to accomplish and that the algorithm is almost *automatic* - i.e. it doesn't require a deep analysis of the potential data that will be stored in the database. The algorithm is very simple and its application is almost immediate.

3. Numerical Model

The object data model formalization is based on one of the many equivalent models in literature [8]. Hierarchical clustering splits a set of items (in our case objects) according to their similarities on a common set of features that can be quantified. We could let the set of features be values of the object attributes or method results but this would lead to a classification based only on attributes. Classifying the objects on their static data would certainly lead to a set of fragments, but those fragments would not express at all the dynamics of the system. This is hardly interesting for a distributed database where data is only the static dimension. The dynamic part is represented by the applications that access attributes and send messages to the objects according to the class protocol. Our quantification of the features of each object will not be value/attribute based but application based. The quantification leads to a numerical representation of each object in a vector space. We classify then objects into clusters according to their similarities in behavior in the context of the running applications.

Let $Class = \{C_i | \, C_i$ is a class in the system$\}$ be the set of all classes. The extension of a class C_i, denoted $Inst(C_i)$ is the set of all instances *(objects)* of that class: $Inst(C_i) = \{O_j | \, O_j$ is an instance of $C_i\}$. We denote by $Q = \{q_1, \ldots, q_t\}$ the set of all queries*(applications)* that will be running in the system. It should be noted that only applications running with some frequency are taken in account for quantification as those are the ones that will be most influenced by the resulting clusters. Considering an SQL based system, each of those applications will have filters of *where clauses* that will filter the accessed objects. Let $Pred = \{p_1, \ldots, p_q\}$ be the set of all atomic predicates Q is defined on. Let $Pred(C) = \{p \in Pred | \, p$ imposes a condition to an attribute of class C or to an attribute of its parent$\}$.

Given the predicate $p : C_1.A_1. \ldots A_n \; \theta value$, $p \in Pred(C_n)$, where class C_i is the complex domain of A_{i-1}, $i = 2..n$, and A_n is an attribute of C_n that has a complex type(another object) or a simple type (scalar). $\theta \in \{<, >, \leq, \geq, =, \neq, \in, \supset, \supseteq\}$ is a filter operator, while $value \in domain(A_n)$

For each object $O_i \in Inst(C_j)$ we can derive a vector having $|Pred(C)|$ dimensions, each corresponding to a predicate and having a value of *one* if the object is selected by that predicate or *zero* otherwise. All object vectors for objects of a class C yield a matrix denoted *object-condition matrix OCM(C)*: $OCM(C) = \{a_{ij}, 1 \leq i \leq |Inst(C)|, 1 \leq j \leq |Pred(C)|\}$, where $Inst(C) = \{O_1, \ldots O_m\}$ and $Pred(C) = \{p_1, \ldots, p_n\}$.

Table 1 shows an example of a object-condition matrix. Each line in the matrix is the object-condition vector of the corresponding class instance. The features (is selected or not) of each object are only qualitative. If we want to integrate some quantitative information about how objects are filtered by a predicate we can add in the percent of objects that are filtered in the same manner by a predicate. Right side of Table 1 shows the CVM.

OCM(C)	**p1**	**p2**	**p3**	**p4**
O_1	1	0	1	1
O_2	0	1	0	1
O_3	1	1	0	0
O_4	0	0	0	0
O_5	1	1	0	0
O_6	0	1	0	0

CVM(C)	**p1**	**p2**	**p3**	**p4**
O_1	0.5	0.33	0.16	0.33
O_2	0.5	0.66	0.84	0.33
O_3	0.5	0.66	0.84	0.66
O_4	0.5	0.33	0.84	0.66
O_5	0.5	0.66	0.84	0.66
O_6	0.5	0.66	0.84	0.66

TABLE 1. Object-condition and characteristic matrix for a class

A new matrix *(characteristic vector matrix)* $CVM(C) = \{w_{ij} | i = \overline{1..m}, j = \overline{1..n}\}$ having same dimensions is obtained and defined as:

$$w_{ij} = \frac{\sum_{l=\overline{1..m}, a_{lj}=a_{ij}} [(a_{lj}|a_{lj} = 1) + (1 - a_{lj}|a_{lj} = 0)]}{m} \tag{1}$$

4. Hierarchical Clustering fragmentation

We obtain a numerical model that expresses the dynamic behavior of the data in the context of user applications. The only missing thing is a measure of similarity/dissimilarity between objects and an algorithm capable to take as input the OCM or VCM matrices and produce the desired clusters by grouping together only similar objects.

The similarity functions we used for our tests are based on some well known metrics (euclidian and manhattan):

$$d_E(we_i, we_j) = \sqrt{\sum_{k=1}^{n} (we_{ik} - we_{jk})^2}, \; d_M(we_i, we_j) = \sum_{k=1}^{n} |we_{ik} - we_{jk}| \tag{2}$$

$$(3)\quad sim_E(O_i,O_j)=1-\frac{d_E(we_i,we_j)}{|Inst(C)|},\ sim_M(O_i,O_j)=1-\frac{d_M(we_i,we_j)}{|Inst(C)|}$$

The similarity functions have values between 0 (meaning that objects are totally dissimilar) and 1 (meaning full similarity). The similarity needs to be bounded so that we could asses 0 and full similarity. The hierarchical clustering algorithm we used is presented bellow. The algorithm starts with $m = |Inst(C)|$ clusters, each containing a single object. The main iteration unifies the two most similar clusters until the number of remaining clusters drops bellow the desired number of clusters.

```
Algorithm HierachicalFragPrimar is
Input:  C-class to cluster, Inst(C) - instances of C, similarity
function sim : Inst(C)xInst(C) ⟶ [0,1], m = |Inst(C)|, k-the number
of desired clusters where 1 < k ≤ m and OCM(C)/CVM(C) − w_ij .
Output:  The set of clusters F = {F_1, ..., F_k}
Begin
    For i=1 To |Inst(C)| do F_i = {w_i};
    F = {F_1, ..., F_m};
    While |F|>k do
       // Find (F_u*, F_v*) with the greatest similarity
       (F_u*, F_v*) := argmax(F_u, F_v)[sim(F_u, F_v)];
       Fnew = F_u* ∪ F_v*;
       F = F − {F_u*, F_v*} ∪ {F_new};
    End While;
End.
```

5. Results

The performance evaluation in the case of fragmentation is generally a complex issue to be dealt with. Normally the generated fragments should be allocated to the nodes of a distributed system. Then the applications are run against the system and various parameters like execution times, data transfer amounts, etc are measured. Since the allocation of the clusters to nodes problem is by itself an entire research subject for which there is no yet a linear solution, one can choose to do a simple allocation schema like: allocate each cluster to the node where it is most used. This is the approach we used in our tests. In order to be able to find the nodes where a cluster is most accessed locally we need to apriori know:

- The objects that the application accesses, for each application and class;
- The number of nodes in the system;
- The frequency of running a given application on a given node;

Given a system with $S = \{S_1, \ldots, S_S\}$ nodes, each application runs with a certain frequency on each node of the system,$freq_{S_j}(q_i)$. We computed the general impact application q_i has on the clustering process as being the sum of all frequencies over all the nodes of the system:

$$freq(q_i) = \sum_{s=1}^{S} freq_{S_s}(q_i)$$

By definition we only consider applications that have $freq(q_i) > 0$. A general frequency, $freq(q_i) = 0$ means that the application is not running in the system - so it is not useful for the clustering process.

We denote by $p_i \in q_j$ the fact that p_i is part of the filters *(where clauses)* that define q_i. We would like to capture the impact predicate p_i has on the clustering process according to its frequency of execution as well. This means we need to weight the OCM/CVM matrices to take into account the frequency. A predicate with a high execution frequency should have larger influence on the clustering process than a predicate with a low execution frequency. Having $Q = \{q_1, \ldots, q_t\}$ and $FreqGen = \{freq_1, freq_2, \ldots, freq_t\}, 0 < freq_i \leq freqQMax, freq_i \in Z$. When a predicate p_j is a filter in more than a single application its frequency is the sum of individual execution frequencies of each application that uses p_j.

$$freq(p_j) = \sum_{i=1, p_j \in q_i}^{t} freq(q_i), 0 < freq(p_j) \leq freqMax \in Z$$

In order to use the predicate frequencies weights in the OCM/CVM matrices we need to shift their values in a well know bounded interval, keeping in the same time their semantic. After the interval shift we can directly weight the OCM/CVM matrices:

$$w_{ij}^{'} = w_{ij} \times \frac{freq(p_j)}{\sum_{i=1, p_j \in q_i}^{t} freqMax} \tag{4}$$

For the numerical evaluation of the proposed model we consider small, medium and large datasets. In order to asses the clustering quality we use a set of applications with a predefined set of frequencies randomly chosen.

We use the partition evaluator proposed in other similar works as [4, 2, 6] for cluster quality:

$$PE(C) = EM^2 + ER^2 \tag{5}$$

The evaluator (PE) computes the cost of accessing local data (EM) and remote data (ER) when running the set of user queries over the fragments of a class. As the value of the cost increases, the quality of fragmentation is lower.

$$EM^2(C) = \sum_{i=1}^{M}\sum_{t=1}^{T} freq_{ts}^2 * |Acc_{it}| * \left(1 - \frac{|Acc_{it}|}{|F_i|}\right) \tag{6}$$

$$ER^2(C) = \sum_{t=1}^{T} \min\left\{\sum_{s=1}^{S}\sum_{i=1}^{M} freq_{ts}^2 * |Acc_{it}| * \frac{|Acc_{it}|}{|F_i|}\right\} \tag{7}$$

The EM term computes the local irrelevant access cost for all fragments of a class. ER calculates the remote relevant access cost for all fragments of a class. Acc_{it} represents the set of objects accessed by query t from fragment F_i. The value $freq_{ts}$ is the frequency of query t running on site s. In (6) s is the site where F_i is located, while in (7) s is any site not containing F_i. M is the number of clusters for class C, T is the number of queries and S is the number of sites. The fragmentation is better when the local irrelevant costs and the remote relevant access costs are smaller. Each term of PE calculates in fact the average square error of these factors.

The quality of fragmentation expressed as the cost of evaluating queries against the resulting database is expressed in Figure 1:

In Figure 1 we compare the application execution costs for a small database clustered with the k-means algorithm (random initial centroids), hierarchical clustering algorithm, single site database, fully replicated database and the fragmented dataset when using the method exposed in [10]-*Bai* and in [9] - *Bel*. According to the PE measure the single node database and full replication both obtain very high, respectively high costs. This is mostly due to ER term scoring very high in the case of single node database - most of the data is accessed remotely in this case. For the fully replicated case the high score is due to the local irrelevant accesses to data (EM).

We apply the hierarchical and k-means clustering to both OCM and CVM matrices. The better results of the hierarchical clustering are due here to the random initial centroids for the k-means method. Normally the k-means algorithm should perform better in these scenarios, but the random choice of the initial centroids often lead to lost clusters and bad results. Overall,

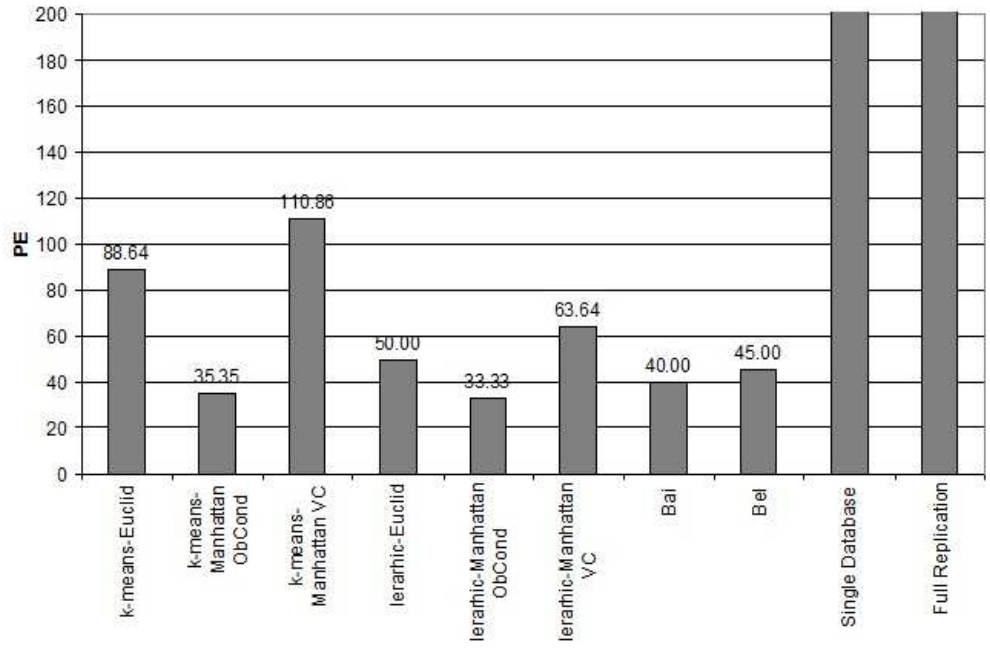

FIGURE 1. Cluster quality evaluation - PE measures.

on small databases the hierarchical clustering performs better than all other methods. The best result is obtained when applying the algorithm on object-condition matrices. The binary selection of predicates yields a better selection than the quantification of *selected/not selected percents* of application objects.

Figure 2 shows the results from a scalability point of view. Three different dataset sizes are tested: small, medium and large datasets. The small dataset context has already been presented in Figure 1. With the significant growth of the dataset, the inefficiency of randomness in centroid choosing fades away and the k-means method takes its place with the smallest cost. Hierarchical clustering scores linearly with the database size, as unifying entire clusters is prone to introducing misplaced objects together with the good ones. At this level hierarchical clustering is too coarse as it does not allow for individual object placing in clusters. Bei and Bel methods are not very affected by the size changes.

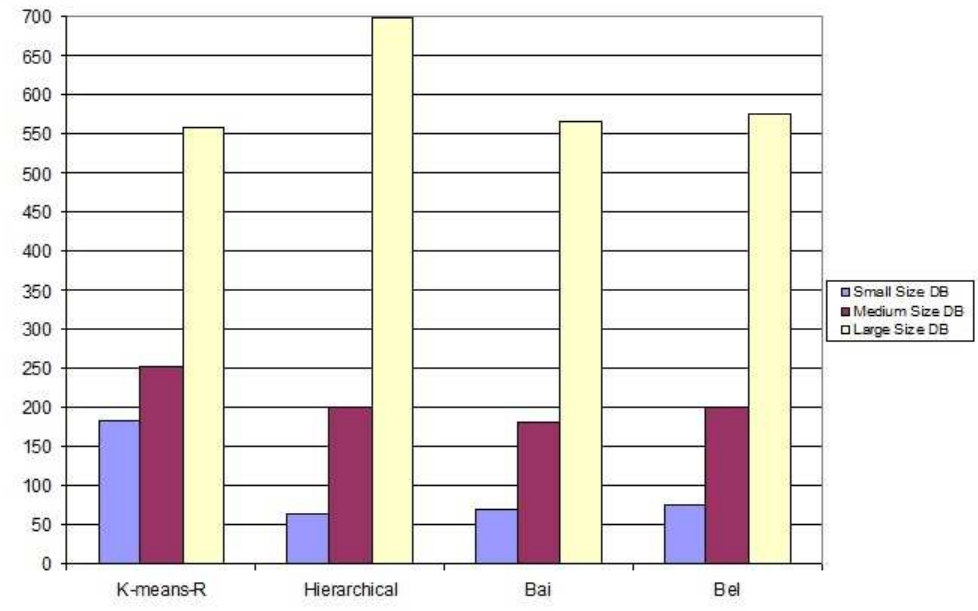

FIGURE 2. Quality evaluation for small, medium and large datasets

6. CONCLUSIONS

Datasets and database fragmentation is a difficult subject to approach from a practical point of view. Existing methods are far too demanding in knowledge and experience for the average DBA to be able to perform them correctly. In this paper we present a dataset fragmentation method based on hierarchical clustering that is easy to apply. Basically, we need as input some quantitative information about the applications that run in the system and the sets of data selected by their filters. Building the OCM/CVM matrices is straightforward and applying the algorithm does not require any additional knowledge from the user. From the performance point of view the algorithm produces clusters comparable in quality with other more elaborate fragmentation methods. We have also shown that our simple method yields good results when evolving the database scale from small to medium.

REFERENCES

[1] Han, J., Kamber, M., Data Mining: Concepts and Techniques, The Morgan Kaufmann Series in Data Management Systems, 2000.

[2] Karlapalem, K., Navathe, S.B., Morsi, M.M.A. - Issues in distribution design of object-oriented databases. In M. Tamer Ozsu, U. Dayal, P. Valduriez, editors, Distributed Object Management, pp 148-164, Morgan Kaufmann Publishers, 1994.
[3] Karlapalem, K., Li, Q., Vieweg, S. - Method Induced Partitioning Schemes in Object-Oriented Databases, In Proceedings of the 16th Int. Conf. on Distributed Computing System (ICDCS'96), pp 377-384, Hong Kong, 1996.
[4] Ezeife, C.I., Barker, K. - A Comprehensive Approach to horizontal Class Fragmentation in a Distributed Object Based System, International Journal of Distributed and Parallel Databases, 33, pp 247-272, 1995.
[5] Karlapalem, K., Li, Q. Partitioning Schemes for Object-Oriented Databases, In Proceedings of the Fifth International Workshop on Research Issues in Data Engineering-Distributed Object Management, pp 42-49, Taiwan, 1995.
[6] Darabant, A. S, Campan, A. - Semi-supervised Learning Techniques: k-means Clustering in OODB Fragmentation, In Proc of the IEEE Intl Conf on Computational Cybernetics ICCC 2004, pag: 333 338, Wien, Austria
[7] Darabant A. S., A new approach in fragmentation of distributed object oriented databases using clustering techniques, in Studia Univ. Babes Bolyai Informatica, Vol I, No 2, pag 91-106, 2005.
[8] Bertino, E., Martino, L. - Object-Oriented Database Systems; Concepts and Architectures, Addison-Wesley, 1993.
[9] Bellatreche,L., Karlapalem, K., Simonet, A. - Horizontal Class Partitioning in Object-Oriented Databases, In Lecture Notes in Computer Science, volume 1308, pp 58-67, Toulouse, France, 1997.
[10] Baiao, F., Mattoso, M. - A Mixed Fragmentation Algorithm for Distributed Object Oriented Databases, In Proc. Of the 9th Int. Conf. on Computing Information, Canada, pp 141-148, 1998.
[11] Tamer, Oszu M., Patrick Valduriez. Principles of Distributed Database Systems, Prentice-Hall, 1998.

Faculty of Mathematics and Computer Science, Babe-s-Bolyai University, Cluj-Napoca, Romania
E-mail address: {dadi,anca}@cs.ubbcluj.ro