# Path based K-means Clustering for RFID Data Sets

**Hong-won Yun**, *Member, KIMICS*

***Abstract***— Massive data are continuously produced with a data rate of over several terabytes every day. These applications need effective clustering algorithms to achieve an overall high performance computation. In this paper, we propose ancestor as cluster center based approach to clustering, the K-means algorithm using ancestor. We modify the K-means algorithm. We present a clustering architecture and a clustering algorithm that minimize of I/Os and show a performance with excellent. In our experimental performance evaluation, we present that our algorithm can improve the I/O speed and the query processing time.

***Index Terms***—Modified K-means, K-means clustering, Clustering, Temporal data model

## I. INTRODUCTION

The retail industry is one of the largest industries worldwide. This industry is facing similar trends those affecting other fields of industry. Also, the industry faces specific challenges such as management of the short shelf-life of grocery goods, strict traceability requirement and deal with a growing number of stock keeping units. The number of daily sales transactions has exploded. Therefore, major retailers are very interested in the intelligent network. RFID and the EPC network are expected to provide them many impacts and benefits. The majority of the retailers place their hopes on new information technologies such as RFID and the EPC network. Many major retailers such as Wal-Mart, Tesco, and 7-Eleven adopt these technologies [1-4].

Successful RFID applications will potentially create terabytes of data. For example, Wal-Mart will generate around 7 terabytes of data every day. We can suppose a retailer with 3,000 stores sells 10,000 items a day per store and each item moves10 times on average before being sold. The applications will generate at least 300 million records per day [1]. The presence of terabyte data requires new data storage structure and processing algorithms to provide fast query processing time. Such data volumes will impose severe strains on existing data management and storage structures and policies.

Manuscript received September 4, 2008; revised November 13, 2008. Hong-won Yun is with the Department of Information Technology, Silla University, Busan, 617-736, Korea (Tel: +82-51-999-5065, Fax: +82-51-999-5657, Email: hwyun@silla.ac.kr)

Numerous storage structures have been proposed to deal with different kinds of spatial data and temporal data, and to efficiently handle specific types of queries [5,10-12]. Clustering is one of storage strategies widely used to fast process specific types of queries. K-means is popular clustering algorithm that has been used in variety of application domains [5-7]. We can develop a modified version that can make use of background knowledge. We have developed a K-means variant that can incorporate background knowledge. We focus on how to minimize the number of disk accesses to trace querying objects.

We present a clustering architecture and an algorithm that minimize the number of page input and output and show a performance study which presents that our algorithm perform well. In the next section, we provide some background on the K-means algorithm and present tree structure for a data management in the retail industry. Then in Section III we propose our modified K-means algorithm. Next, we present experimental results and compare our work to related research in Section IV. Finally, Section V summarizes our contributions.

## II. RELATED WORK

### *A. K-means Clustering*

Clustering is an important subject of application for a variety of fields, such as data analysis, information retrieval and image segmentation. The fundamental clustering problem is that of grouping together data items that are similar to each other. The K-means clustering has been shown to be effective clustering results for many practical applications. We briefly describe the K-means clustering algorithm. K-means clustering is a method commonly used to automatically partition a data set into $k$ groups [6,8,9].

This algorithm proceeds by selecting $k$ initial cluster centers and iteratively refining them. The number of clusters k is assumed to be fixed in K-means clustering. The selection of $k$ is key problem and domain dependent. A user can choose several values of $k$. The K-means algorithm can be decomposed into two parts as follows [6]:

1. Each data item $C_j$ is assigned to its closest cluster center (centroid).
2. Each data item $C_j$ is updated to be the mean of its constituent data.

Below algorithm illustrates a high level description of the K-means clustering algorithm [8].

K-means()
Initialize $k$ prototypes $(\rho_1,\ldots,\rho_k)$ such that $\rho_j = i_l, j\in\{1,\ldots,k\}$, $l\in\{1,\ldots,n\}$
Each cluster $C_j$ is associated with prototype $\rho_j$
*do*
*for* each input vector $\rho_j$, where $l\in\{1,\ldots,n\}$,
*do* assign $i_l$ to the cluster $C_j$ with nearest prototype $\rho_j$.
*for* each cluster $\rho_j$, where $j\in\{1,\ldots,k\}$,
do update the prototype $\rho_j$ to be the centroid of all objects in $C_j$, so that

$$\rho_j = \sum_{il\in Cj} il/|Cj|$$

compute the error function:

$$F = \sum_{j=1}^{k} \sum_{i_l \in C_j} |i_l - \rho_j|^2$$

*while* $F$ does not changes or cluster membership no longer changes

The K-means is defined over numeric data since it needs the ability to calculate the mean. There are various approaches to determine optimal values of the parameters given the data. We focus on the determination of the centroids to be clustering. In order to facilitate query processing we will modify the K-means algorithm.

### *B. Stages of object movements*

In [1] Hector Gonzalez et al. present several general idea for constructing a highly compact RFID data warehouse. Almost of objects move and stay together the supply chain with several stages. If 1000 packs of product P stay together at the distribution center, register a single record. Gid [1] is a generalized identifier that represent the 1000 packs that stayed together at the distribution center. In this paper, we use *Gid* to reduce the number of records and stages of object movements as shown in Fig. 1.

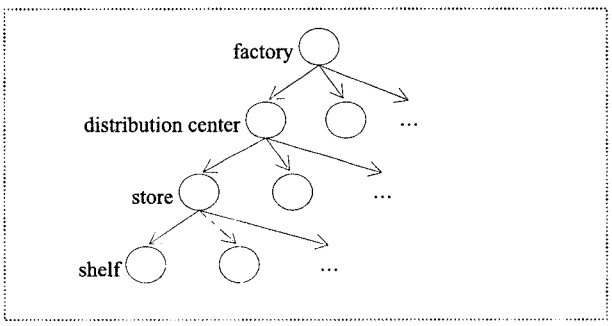


Fig. 1 A sample of object movements

## III. PATH BASED CLUSTERING

### *A. Data model and basic concept*

We consider Fig. 1 is basic sample to construct our data model. This tree structure has an ancestor and many child nodes. One package register a single record with *gid* is root node and it is modeled as $T\alpha = (R, V, E, L)$, where $R$ is a root node (ancestor). $V$ is the set of nodes which are linked by edges $E$. $L$ is a label that is assigned in each node $V$. In Fig. 2, l, we are given a set of k parent objects and just one ancestor denoted 1. The intuitive idea for clustering is that one ancestor with child nodes construct a cluster whose centroid is the ancestor. If one ancestor has large volume of data exceed the critical value, then a cluster is partitioned into several numbers of clusters. Each parent node becomes a centroid in the new cluster respectively.

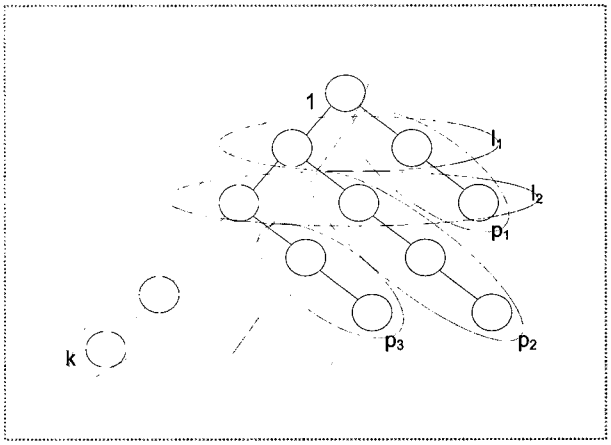


Fig. 2 Same path, same level, and same parent

Fig. 2 say that ellipse notation $l_1$, $l_2$, …, $l_i$ mean with same level and $p_1$, $p_2$,…, $p_j$ present with same parent. The main intuition behind our technique is as follows. All the nodes with same path are potential candidate for the nearest a membership of cluster at the ancestor with centroid. Same level nodes are given a set of nodes may belong to the candidate for a membership of cluster secondly at the same ancestor.

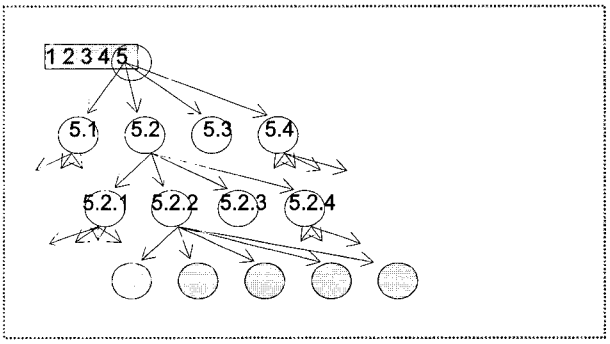


Fig. 3 Labeled node with same path and same level

We will assign label to each node to discriminate. The label will contain one identifier that is numbered from one ancestor. For example, Fig. 3 shows the label for each node from the ancestor. We see that one node with label 5, and then subdivide into 5.2, and subdivide further into 5.2.2. The information of same path 5→5.2→5.2.2→5.2.2 is stored at attribute named Gid in the table as shown in Fig. 4. Temporal information is also important together path information to monitor and trace items. There are two different approaches to incorporating a temporal dimension into the relational

data model such as the temporally grouped and the temporally ungrouped approaches. Fig. 4 presents the temporally grouped (TU) data model and the temporally ungrouped (TG) data model as shown in Fig. 5.

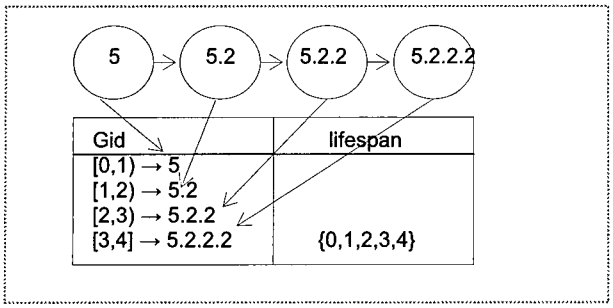


Fig. 4 Temporally grouped data model

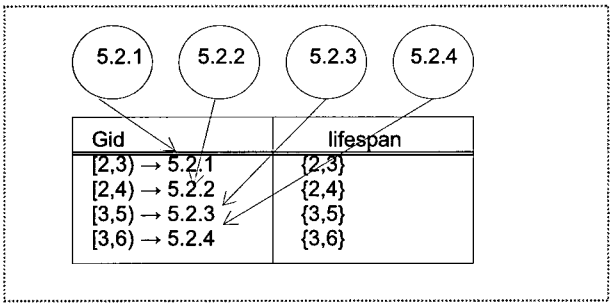


Fig. 5 Temporally ungrouped data model

You can have a choice either one as logical data model. The user can always just add whatever extra time attributes are desired and then use stand SQL as the query language. In the figure we just see that it is possible to adopt TU data model or TG data model for historical querying

### *B. Clustering algorithm using ancestor*

The appropriate choice of centroid is key problem. In this paper, each ancestor is centroid that originally meaning is a product package. Each ancestor and child nodes have Gid respectively. Each item with Gid as an identifier is grouped same level, same path, and same parent as shown in Fig. 6.

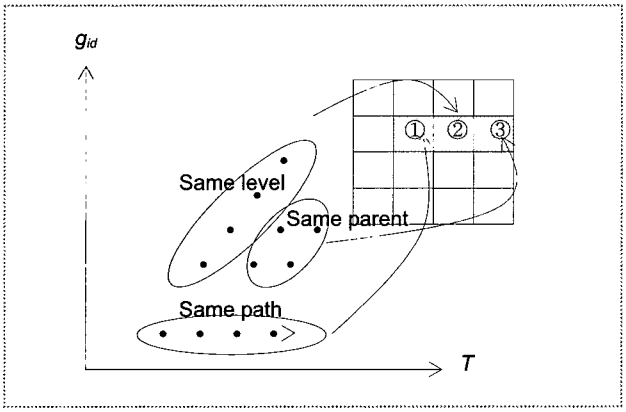


Fig. 6 The movement of same path and same level to the centroid

The nodes of same path move to closest space from centroid firstly, and secondly the nodes of same level move to it. In Fig. 6, the ① shows the first priority to move, ② is the second, and ③ is third. This assignment reduces distance from centroid as its ancestor to each node in the same path, and then gets fast response. The true cluster means are depicted by "+" as shown in Fig. 7.

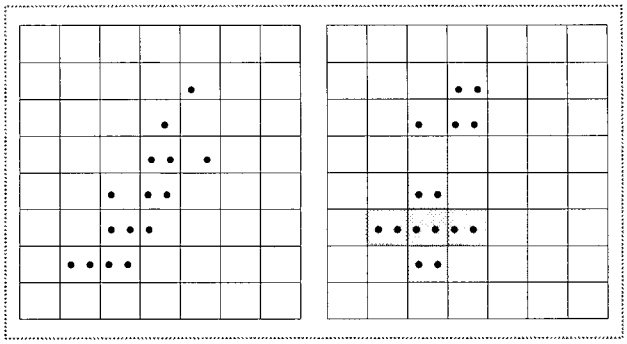

Fig. 7 Result of clustering using ancestor

The main intuition of our algorithm is that the ancestor is potential candidate for the centroid at the root level. We need basic information for each node of the tree structure, we keep the following information:

1. The point of ancestors ($p$)
2. The point of children of the ancestor($sa$)
3. The points of same path with the ancestor($sp$)
4. The points of same level with the ancestor($sl$)

Our algorithm takes as input: initial centroids, data set, and the point of ancestor:

**Algorithm Clustering_with_SA**

1. Let $C_1, \ldots, C_i$ be the initial centroids.
2. Let $S_i$ be a random cluster of data set.
3. For each cluster $S_i$, update its centeroid by computing all of the points $p$.
   a. For each $P_i \in S_i$: $AP_i$ = K-means($S_i$, $C_i$, $sp_i$)
   b. $AP = AP \cup AP_i$
   c. For each $P_i \in S_i$: $BP_i$ = K-means($S_i$, $C_i$, $sl_i$)
   d. $BP = BP \cup BP_i$
4. For each $P_i$, Let $CP = AP \cup BP$
5. Iterate between a, b of (3) and c, d of (4) until end of data set.
6. Return $\{CP_1, \ldots, CP_k\}$

## IV. EXPERIMENTAL RESULTS

In this section, we present some experiments we performed to assess the effectiveness of the proposed method in large volume of data. We have compared our algorithm with K-means algorithm in terms of the number of page I/Os, processing time for queries and the number of I/O operations. For our tests, we used experimental data that a record size is 256 byte and 4 kilobyte page size. Also, we varied the number of queries, rate of object tracking queries, and number of

records. The experiments were performed with randomly generated data having a uniform distribution.

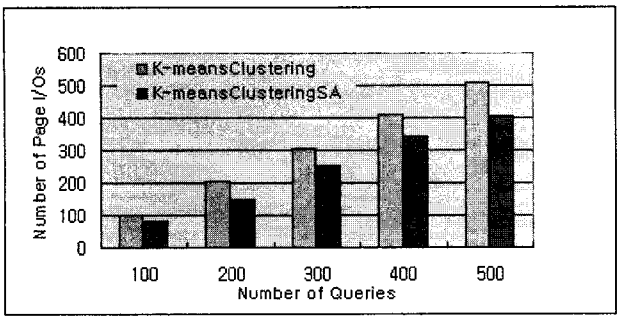


Fig. 8 Number of page I/Os

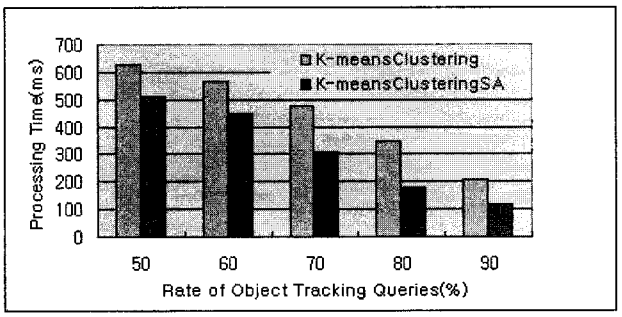


Fig. 9 Query processing time

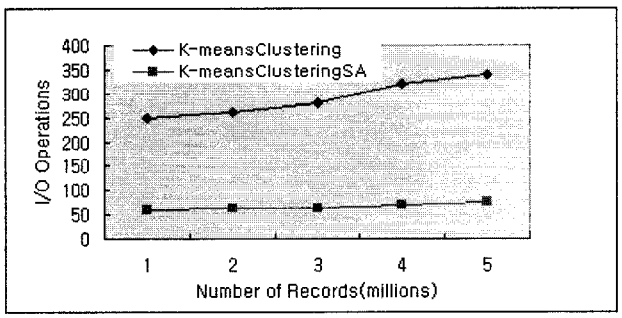


Fig. 10 I/O operations

The results of our experiment over varying the number of queries between 100 and 500 are shown in Fig. 8. In this experiment, K-means clustering using ancestor has the lower number of page I/Os than K-means. The difference of number of page I/Os is increasing according to number of queries. The results of query processing time over varying rate of object tracking queries are shown in Fig. 9. The results presented in Fig. 9 show that our algorithm result in better improvement in overall rate of object tracking queries than K-means. At rate of object tracking queries=90, the K-means using ancestor is performing highly better than K-means. I/O operations are increasing according to varying number of records as shown in Fig. 10. When there are 1 millions or more, the I/O operation using K-means with ancestor is better than the K-means clustering. When there are 3 millions or more, K-means clustering is I/O intensive, on the other hand, our proposed algorithm is stable. We can conclude that the K-means using ancestor provides performance with excellent overall execution time. Through experiments our K-means using ancestor are expected to be more efficient, resulting in better performance.

## V. CONCLUSIONS

The number of daily sales transactions has exploded in the retail industry. Many major retailers adopt RFID applications. These applications continuously produced with a data rate of over several terabytes every day. These systems need effective clustering algorithms to achieve an overall high performance computation. Clustering is one of storage strategies widely used to fast process specific types of queries. K-means is popular clustering algorithm that has been used in variety of application domains. We have developed a K-means variant that focus on how to minimize the number of disk accesses to trace querying objects. In this paper, we proposed ancestor based approach to clustering. We modified the K-means algorithm and present a clustering architecture. The clustering algorithm minimized of I/Os and show a performance with excellent. In our experimental performance evaluation, we present that our algorithm can improve the input output speed and the query processing time.

## REFERENCES

[1] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and Analyzing Massive RFID Data Sets," 22nd IEEE ICDE Conference, 2006, p.1.

[2] Fosso Wamba et al., "Enabling Intelligent B-to-B eCommerce Supply Chain Management using RFID and the EPC Network: a Case Study in the Retail Industry," *International Journal of Networking and Virtural Organizations*, 3(4), 2006, pp. 450-462.

[3] E. Masciari, "RFID Data Management for Effective Objects Tracking," Proceedings of the 2007 ACM sysposium on Applied computing, 2007, pp. 457-461.

[4] L. Golab and M. Tamer Ozsu, "Issues in Data Stream Management," SIGMOD Record, Vol. 32, No. 2, 2003, pp. 5-14.

[5] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," ACM Computing Surveys, Vol. 31, No. 3, 1999, pp. 264-323.

[6] K. Wagstaff and S. Rogers, "Constrained K-means Clustering with Background Knowledge," Proceeding of the Eighteenth ICML, 2001, pp. 577-584.

[7] H. Nagesh, S. Goil, and A. Choudhary, "Adaptive Grids for Clustering Massive Data Sets," Proceeding of the 1st SIAM ICDM, 2001, pp. 1-17.

[8] K. Alsabti, S. Ranka, and V. Singh, "An Efficient K-Means Clustering Algorithm," First Workshop on High-Performance Data Mining, 1988, pp. 1-6.

[9] P.S. Bradley and U.M. Fayyad, "Refining Initial Points for K-Means Clustering," Microsoft Research TR, pp. 0-9.

[10] S. Nittel, K.T. Leung, and A. Braverman, "Scaling Clustering Algorithm for Massive Data Sets using Data Streams," Proceedings of the 19th ICDM, 2003.
[11] A.A. Diwn, S. Rane, S. Seshadri, and S. Sudarshan, "Clusterig Techniques for Minimizing External Path Length," Proceedings of the 22nd VLDB Conference, 1996, pp. 342-353.
[12] F. Wang and P. Liu, "Temporal Management of RFID data," Proceeding of the VLDB05, 2005, pp.1128-1139.

**Hong-won Yun**
He received his B.S. and the Ph.D. degrees at the Department of Computer Science from Pusan National University, Korea, in 1986 and 1998, respectively. He is a professor at the Department of Information Technology, Silla University in Korea. His research interests include temporal database and semantic web.