Toward Optimal Replication for Hierarchical Location Management in Wireless Systems*

San-Yih Hwang and Jeng-Kuen Chiu

Department of Information Management, National Sun Yat-Sen Univeristy, Kaohsiung, Taiwan 80424
syhwang@mis.nsysu.edu.tw, jkchiu@ms17.hinet.net

Abstract. The location information of users can be replicated at various databases in a hierarchy to improve the efficiency of call lookups at the expense of increase in update and storage cost. In this paper, we systematically investigate this problem and propose two solutions. The first solution has optimal lookup and update cost, but the execution time could be exponential to the number of databases. The second solution makes some assumptions and then uses dynamic programming to tackle this problem. Its execution time is dramatically reduced with the result being less optimal. Finally, to further improve efficiency and reducing storage requirements, we propose the incorporation of clustering techniques.

1 Introduction

Establishing a connection in a wireless environment requires the location information of the callee, who may be on the move. While this problem is mostly encountered at the data link or networking layer transparently from the layers above it, applications may also need the location information at times, for instance, to contact or move a mobile agent, to answer queries that involve locations, or to update environmental parameters. This implies that future personal communication services (PCS), with high user populations and numerous customer services, will incur heavy signaling and database traffic for locating users [16]. Thus, deriving efficient strategies for location management is an important issue to mobile computing research.

In essential, there are two primitive operations for location management: lookups, or searches, for the current location of a user when a call to her/him is required, and updates when a user moves to a new location. Two kinds of architectures for location databases are widely discussed in existing researches: the two-tier architectures based on a pair of HLR/VLR and the hierarchical architectures composed of a number of location registers connected through the intelligent network, such as SS7. In the following, we extract the significant approaches that have been proposed to reduce the cost of lookups and updates in both architectures [13].

* This research has been supported by the National Science Council, ROC, under grant NSC 87-2213-E-110-010.

1.1 HLR/VLR schemes

In the basic two-tier scheme, as applied in IS-41 and GSM systems [10], two types of databases, namely *Home Location Register* (HLR) and *Visitor Location Register* (VLR), are maintained for storing the subscribers' location information. Each mobile system is associated with a HLR, which contains the location information of users who subscribe to the system. As a mobile system may cover numerous smaller geographical areas called zones, each zone is equipped with a VLR, which stores location information of those users who currently visit the zone. When a call is originated, the VLR of the caller is always queried first before consulting the HLR of the callee. When a user moves across zones, the entry for the user is deleted from the old VLR in addition to updating the user's record in the HLR, and a new entry for the user is created in the VLR of the newly arrived zone.

To provide faster lookup, [4] and [15] propose to *cache* or *replicate* the location information of a user in the databases of additional zones, so that subsequent calls from these zones can reuse this information. User *i*'s location information is dynamically determined to store in the database of zone *j* if her Local Call to Mobility Ratio $LCMR_{i,j}$ is larger than a threshold [2].

Rather than dynamically deciding where to cache a user's location information, [15] suggests a static way in determining where to replicate the location information, given that the call rates and mobile rates of each user will not make abrupt change between the adjacent invocations of their algorithm. In this case, the location information of a user is replicated at some databases only if these replications bring more savings on the lookup cost than the increase on the update cost. It has been shown that off-line replication assignment problem can be reduced to a maximum-flow problem of a network.

While the previous two schemes help to reduce the lookup cost for those who receive more calls but do less move, those who receive calls less frequently relative to their moving rate do not benefit. As opposed to caching or replication, *forwarding pointers* is proposed with the aim to reduce the update cost [5]. In this scheme, each time a mobile user moves to a new location, a forwarding pointer is installed on its old VLR to point to the new VLR without the change to its HLR. Thus subsequent consulting to the HLR will refer to the first VLR at which the user was registered, and a chain of forwarding pointers to the user's current VLR is followed.

1.2 Hierarchical schemes

To improve scalability, hierarchical location management extends the two-tier schemes by maintaining a hierarchy of location databases. In its basic form, a leaf location database contains entries for all users registered in its cell, and the one at a higher level summarizes the location information (pointers) for users located at levels below it. When it comes to call a user currently located in zone *i* from zone *j*, the databases along the path from zone *j* to *i* via their least common ancestor (LCA) are all traversed.

As in the HLR/VLR schemes, the ideas of pointer forwarding, caching, and replication can also be applied in variant hierarchical schemes [6,7,9,12]. In the following, we will describe the replication selection in the hierarchical scheme in more detail as it is the main focus of this paper.

Similar to the strategy used in the 2-tier schemes, replicating the location information of users at additional nodes in a hierarchy is allowed only when it is judicious, that is, the benefit of replication exceeds its cost. In this context, replication of the location information may occur at leaf nodes as well as the nodes at higher levels in the hierarchy. At each node, two numbers, R_{min} and R_{max}, where $R_{min} < R_{max}$, are derived. The call to mobility ratio of user *i* at zone *j*, $LCMR_{i,j}$, is measured such that if $LCMR_{ij} < R_{min}$, replication of user *i*'s location information should never be done at database *j*, and if $LCMR_{ij} \geqq R_{max}$, such replication should be always conducted. In case $R_{min} \leqq LCMR_{ij} < R_{max}$, there is no clear conclusion whether the replication of user *i*'s location information at database *j* should be performed. In [7], an off-line algorithm is proposed to determine the sites for replicating the profile of a user *i*. This algorithm proceeds in two phases. In the first phase, all databases are traversed in a bottom-up fashion, and replicas of user *i*'s profile is stored in a database *j* only if $LCMR_{ij} \geqq R_{max}$. As there is a limit on the maximum number of replicas a user profile can have, the second phase allocates the remaining replicas to databases with the largest non negative $LCMR_{ij} - R_{min}$, in a top-down fashion.

1.3 Motivation and Paper Organization

This paper focuses on the hierarchical scheme with user profile replication. The two-phase algorithm proposed in [7], though simple, does not provide insights on whether or why it works well. We discuss the nature of the replica assignment problem in the context and propose an optimal solution to it. As the optimal solution takes a long time to compute, we make further assumptions to simplify the problem and then solve it via dynamic programming. Finally, rather than determining the replica assignment on a per-user basis, we propose to first cluster mobile users based on their calling and moving patterns and then perform the replica assignment for each group. This will further improve the efficiency of replica assignment, in addition to reducing the storage requirements.

The remainder of this paper is organized as follows. In Section 2, we describe the nature of the problem and show how to achieve the optimal placement of profile replicas for each user in terms of the cost of updates and queries to the location information. Motivated by the high complexity of the optimal solution, in Section 3 we offer an approximate solution through a dynamic programming approach. In Section 4, we present the idea to incorporate off-line clustering technique to further reduce the complexity. Section 5 summarizes the current results and points out our future work.

2 The Replica Assignment Problem

We follow the general hierarchical model as proposed in [7]. The problem is to assign the profile replicas to a number of databases such that the overhead incurred due to calling and moving is minimized. To simplify the problem, we consider the system cost, which involves both communication cost and database operating cost, as the primary performance metric. Specifically, minimizing the total system cost incurred during a time unit is our goal. Deciding which databases to store the replicas of a user's profile has to take into account a number of factors. To capture these factors, we decide on several parameters. The notations of these parameters and their meaning are summarized in Table 1.

Table 1. Notations and meaning of parameters

Notes	Meaning
$C_{i,j}$	Number of calls to user *i* from zone *j* during a time unit.
M_I	Number of moves (across zones) of user *i* during a time unit.
B_l	Look up cost.
b_u	Update cost.
N	Number of maximum profile replicas of each user.
K	Number of total databases in the hierarchy.
K'	Number of databases in leaf level of the hierarchy.
D	Number of children of each non-leaf node in the hierarchy.
L	Number of levels in the hierarchy.

Our goal is to find the replica assignment with the minimum cost for each user. Thus, a straightforward approach is to first enumerate all possible assignments. And, for each assignment, we calculate its total cost during a time unit. The assignment with the minimum total cost is finally identified. For a given replica assignment of user *i*'s profile, the total cost during a time unit, *g*, can be calculated as follows:

$$g = \Sigma_{j\in \text{ leaves of the hierarchy}} C_{ij} * hops_j * b_l + M_i * b_u * N', \tag{1}$$

where $hops_j$ is the number of hops it requires to get the location information for a call (to user *i*) from the zone of database *j*, and *N'* the number of replicas in the assignment. The first term in *g* is the total communication cost caused by looking up the user's profile for the calls from every zone, and the second term is that for updating the user's profile due to her/his moving. $Hops_j$ can be determined by the given replica assignment and the way pointers are organized. Let us first consider the pointer organization used in the basic hierarchical scheme. Suppose among a set of profile replicas one copy is assigned as the primary profile. A pointer in a non-leaf database always points to the next lower level database that stores either the primary profile or the pointer to the next lower level database. We can recursively define $hops_j$ as follows:

$$\begin{aligned} hops_j &= 0 \text{ , if database } j \text{ stores the user's profile,} \\ &= L \text{ , if database } j \text{ is the root of the hierarchy and does} \\ &\qquad \text{not store the user's profile,} \\ &= 1 + hops_{parent(j)} \text{ , otherwise.} \end{aligned} \tag{2}$$

In a given replica assignment, let x_j denote whether user *i*'s profile is replicated in database *j*. Solving the above recurrence equation, we obtain

$$hops_j = \Sigma_{l=0 \text{ to } L-1} \prod_{k=0 \text{ to } l} (1 - x_{parent^k(j)}) + \prod_{k=0 \text{ to } L} (1 - x_{parent^k(j)})\, L, \quad \mathbf{(3)}$$

where $parent^k(j)$ denotes the ancestor of database j that is k level higher in the hierarchy. The complexity of computing $hops_j$ is $O(L^2)$. It follows that computing g for an assignment takes $O(K'L^2)$.

The number of replica assignments is equivalent to that of combinations for choosing at most N databases out of K databases, which is

$$C(K, N) + C(K, N-1) + \ldots + C(K,0) = \Sigma_{y=0 \text{ to } N} C(K, y). \quad \mathbf{(4)}$$

When $N=K$, the complexity becomes $O(2^K)$, exponential to K, the total number of databases. If the basic pointer organization is employed, the total time complexity for the above brute force approach becomes $O(\Sigma_{y=0 \text{ to } N} C(K,y) K'L^2)$. It is possible to design another pointer organization that shortens the search path. Since the way pointers are organized will not affect the choice of replication assignment, we will not further discuss it in this paper.

3 A Dynamic Programming Approach

We consider in this section a simpler approach in deciding where to replicate a given user's profile. Let RA(z, K) be the optimal replica assignment problem in the hierarchy rooted at z such that the number of replicas is no more than K and the total benefit of replicating becomes maximum. RA(z, K) achieves the maximum replication benefit of all databases rooted at z. The benefit of replicating the profile at the database j can be defined as the difference between the decrease of total lookup time and the increase of update time during a time unit. The total lookup time is in turn determined by the number of hops a lookup takes before reaching the database that stores the profile. Consider a call placed from the zone of a leaf k in the sub-tree rooted at database j. If any database along the path between k and j replicates the location information, the replication in database j does not help at all. By contrast, if none of them ever replicates the location information, the amount of benefit depends on where the profile is replicated along the path between j and the root. Therefore, the decrease of total lookup time because of database j's replication depends on the replication of databases of its descendants as well as its ancestors. This makes the principle of optimality non-applicable, and we are unable to express this problem recursively for dynamic programming.

We therefore assume the average number of hops a call takes to be a constant, denoted as H, the same assumption made in [7]. Let $level(j)$ be the level of database j in the hierarchy. For each call made from a leaf k in the sub-tree rooted at database j, if none of databases along the path from k to j ever replicates the location information, the decrease of total hops due to the replication on database j is $H - level(j)$. We then try to quantify those calls whose lookups require the traversal to database j. Let $C_{i,j}$ be the number of these calls. Clearly, $C_{i,j} = \Sigma_{k \in D(j)} C_{i,k}(1-X_{i,k})$, where $D(j)$ denotes the (direct) children of j, and $X_{i,k} = 1$ if database k contains the location information of user i and $X_{i,k} = 0$ otherwise. The benefit of replicating user i's profile at database j , B_{ij}, becomes

$$C_{i,j} * b_l * (H - level(j)) - M_i * b_u. \qquad \textbf{(5)}$$

Let $O_{i,j}$ be an optimal solution for $X_{i,k}$. If $O_{i,j}$=0, then all proper descendants of j must constitute an optimal solution for the problem RA$^*(j_1, N_1)$, RA$^*(j_2, N_2)$, …, and RA$^*(j_d, N_d)$, where $j_i \in D(j)$ and $\Sigma_{i=1}{}^d N_i = N$. However, if $O_{i,j}$ = 1, then all proper descendants of j may *not* constitute an optimal solution for the problem RA$^*(j_1, N_1)$, RA$^*(j_2, N_2)$, …, and RA$^*(j_d, N_d)$, where $j_i \in D(j)$ and $\Sigma_{i=1}{}^d N_i = N$-1. Let $g_j(y)$ be the value of an optimal solution to RA$^*(j, y)$. However, although the principle of optimality does not hold in general, we can still define a function $g_r(N)$ as the value of a suboptimal solution to RA$^*(r, N)$, which is defined as follws:

if $O_{i,j}$= 0, **(6)**

$$g_j(n) = \max\{ \Sigma_{k\in D(j)}\, g_k(n_k) : 0 \le n_k \le N \text{ for } k\in D(j), \text{ and } \Sigma_{k\in D(j)}\, n_k = N \},$$

and if $O_{i,j}$= 1, then

$$g_j(n) = \max\{ b_j + \Sigma_{k\in D(j)}\, g_k(n_k) : 0 \le n_k < N \text{ for } k\in D(j), \text{ and } \quad 1+\Sigma_{k\in D(j)}\, n_k = N \},$$

where b_j is benefit of replicating the location information in database j given that all of j's children are optimally assigned its replicas. We now can use dynamic programming to compute the benefit $g_j(y)$ of all database j, y=0, …, N. The algorithm is shown below:

```
(* L is the level of the root in the hierarchy *)
(* Given y is the number of maximum replicas, *)
(* x$_j$[y] is the 0/1 assignment to database j ,*)
(*g$_j$[y] is the benefit of the hierarchy rooted at j , and *)
(*c$_j$[y) is the number of calls that are from the zone of database j
  and cannot find the user profile before reaching database j *)
For each database j in level 0 do
 Begin
   Benefit = C$_{ij}$ * bl * H – Mi * bu.;
   g$_j$[0] = 0; x$_j$[0] = 0;
   if Benefit > 0 then (* user i’s profile is replicated
                          on j for 1 <= y <= N *)
      For y=1 to N do
       Begin
         x$_j$[y] = 1;  g$_j$[y] = Benefit;  c$_j$[y] = 0;
       End
   else (* user i’s profile is not replicated on j *)
      For y=1 to N do
       Begin
         x$_j$[y] = 0;  g$_j$[y] = 0;  c$_j$[y] = C$_{ij}$;
       End
 End
For l = 1 to L do
 Begin
   For each database j in level l do
      For y=0 to N do
       Begin
          Max0 = 0;
          For each combination (y1, y2, …, yd) such
            that $\Sigma$yi = y do
              Begin
                 Child_Benefit = 0;
                 For each child ci of j do
                  Child_Benefit=Child_Benefit+g$_{ci}$(y$_i$);
                    If Child_Benefit > Max0 then
                     Begin
                        Max0 = Child_Benefit;
                         Call0 = 0;
                        For each child ci of j do
                          Call0= Call0 + $c_{ci}$[yi];
                     End
              End
            Max1 = 0;
            For each combination (y1, y2, …, yd) such
            that $\Sigma$yi = y-1 do
              Begin
                Child_Benefit = 0;
                For each child ci of j do
                  Child_Benefit=Child_Benefit+$g_{ci}$($y_i$);
                If Child_Benefit > Max1 then
                  Begin
                    Max1 = Child_Benefit;
                    Call1 = 0;
                    For each child ci of j do
                     Call1= Call1 + $c_{ci}$[yi];
                  End
             End
            Benefit_j = Call1*bl*(H–l) – Mi*bu;
            If Max1 + Benefit_j > Max0 then
              Begin
                 $c_j$[y] = 0;
                 $g_j$[y] = Max1 + Benefit_j;
                 $x_j$[y] = 1;
              End
            else Begin
                $C_j$[y]=Call0;  $g_j$[y]=Max0;  $x_j$[y]=0;
             End
      End
 End
```

Note that the number of all permutations (y_1, y_2, … y_d) such that $\Sigma y_i = y$ is C(y+d, d-1). Thus, the time complexity for the above algorithm is O($K \cdot \Sigma_{y=0 \text{ to } N}$C($y$+$d$, d-1)), which is O(K · C(N+d+1, d)) ≈ O($K \cdot N^d$). This complexity is much better than that of the brute-force approach shown in the Section 2.

However, both the dynamic programming approach and *HiPER* proposed in [7] do not always return optimal replica assignment. In the following section, we present a preliminary experimental result that shows that, in most cases, the replication assignment returned by the proposed dynamic programming approach is significantly better than *HiPER* in terms of the total cost.

4 Experiment

In this section, we present the performance evaluation of three algorithms: the *HiPER* algorithm proposed in [7], the dynamic programming algorithm described in Section 3, and the brute-force approach discussed in Section 2. *HiPER* algorithm is compared because it is also based on hierarchical database model. The result returned by the brute-force approach, which is optimal, serves as a baseline for comparison. The performance metric, as defined in section 2, is the total cost incurred per unit of time. Most parameters listed in Table 1 are directly used in the experiments except C_{ij} and M_i, which model users' calling and moving patterns respectively. In the following, we will describe how these patterns are modelled in our experiment.

We assume the total number of calls each receives during a time unit is normally distributed as N(*CallM*, *CallD*). For each user, we further assume the number of calls she receives from various zones to be a Zeta distribution [7], which is shown in Fig. 1. The sequence of zones in the x-axis is randomly generated.

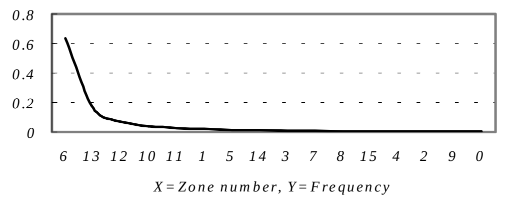

Fig. 1. An example Zeta distribution (*Alpha*=1) of users' call patterns

The average moving frequency of each user is modelled as a normal distribution N(*MoveM*, *MoveD*). Also note that we use *AvgH* to indicate the average number of hops before one profile copy is found, which was defined as *H* in the dynamic programming approach and 2E[*LCA*] in *HiPER*. These parameters and their settings are summarized in Table 2.

Table 2. The setting of parameters

Notes	Setting values
D	4
L	3
N	15
CallM	100
CallD	25
Alpha	0.5 ~ 1.5
MoveM	4
MoveD	2
B_l	1
B_u	1
AvgH	2, 3, 4
HL	1, 2

Fig. 2 compares the cost of three algorithms under *AvgH*=4, *HL*=2. It shows that the dynamic programming approach, *DynaPro,* incurs less cost than *HiPER* in most cases. Besides, the result of *DynaPro* is quite stable and close to the optimal solution. We have also tried other values for *AvgH* and *HL*, and the relative performance between *DynaPro* and *HiPER* remains approximately the same.

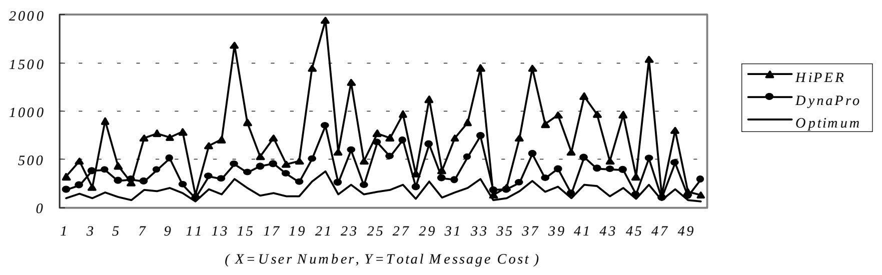

Fig. 2. The performance comparison of three algorithms

5 Incorporating Clustering Techniques

As the area covered by the hierarchy is usually huge, the number of users could be quite large. Deciding the replicas on a per-user basis is thus very time-consuming. We notice that some users could have similar behavior in their calling and moving patterns, especially when bigger zones, as covered by the databases at higher levels in the hierarchy, are considered. This section describes an approach that incorporates clustering techniques in determining the replica assignment.

Clustering techniques arise naturally in several ways where the process to create partitions or clusters is required. To convey clustering to computers, the vague concept of association for partitioning in human behavior must be translated into a numeric measure of the degree of similarity or dissimilarity [1]. Various suggestions to measure the similarity between objects of various types have been proposed [14]. In practice, the Minkowski metric (L_q), as described below, is used to measure the similarity between two interval or ratio vectors, with the distances of Manhattan (q=1) and Euclidean (q=2) as special cases. That is, the distance between vectors $<X_{i1}, X_{i2}, \ldots, X_{ip}>$ and $<X_{j1}, X_{j2}, \ldots, X_{jp}>$ in p-dimensional space is:

$$L_q=(|X_{i1}-X_{j1}|^q+|X_{i2}-X_{j2}|^q+|X_{i3}-X_{j3}|^q+\ldots+|X_{ip}-X_{jp}|^q)^{1/q} \quad (7)$$

Based on the measure of similarity, existing algorithms for clustering can be classified into two categories: hierarchical clustering and partitioning clustering [8]. The former organizes objects as a nested sequence of groups. An important characteristic of this method is visual impact of the dendrogram, which enables a data analyst to see how objects are to be merged into clusters or splitted at successive levels of proximity. Thus, the analyst can try to decide how many clusters to be generated at some fixed level of proximity, which makes most sense for the application in hand. It can be progressed either agglomerative or divisive in the variant linkage methods [3].

Given the number of partitions *k*, a partitioning method tries to find the best *k* partitions. It attempts to determine a cluster in which the objects are more similar to each other than in another clusters. There are many clustering techniques based on this kind of approaches, such as K-means, K-medoid and fuzzy analysis [1], [11]. It has been shown that the clusters produced in partitioning methods depend on the initial values for the means and group numbers [8]. To avoid this weakness, two-phase clustering has been suggested to take advantage of both schemes, as will be described below.

Our work tries to cluster mobile users according to their mobile patterns. This is motivated by the inefficient computation to decide the profile replicas for each user. After clustering users into groups, profile replication can be conducted for each group. This will improve the execution efficiency as well as reducing the storage requirements. In this scenario, each member of the same group replicates her/his profile in the same databases. To perform clustering, we need to define the vector for each user and the similarity function. Call to mobility ratio (LCMR) at each zone is adopted as the basic element for constituting vectors. Specifically, each user is represented as a vector composed of a sequence of LCMRs, each of which is for the zone in a particular level in the hierarchy. The Minkowski distance (L_q), as defined below, has been adopted as the similarity function:

$$L_q = (|LCMR_{i1}\text{-}LCMR_{j1}|^q + |LCMR_{i2}\text{-}LCMR_{j2}|^q + \ldots + |LCMR_{ik'}\text{-}LCMR_{jk'}|^q)^{1/q} \qquad (\mathbf{8})$$

Our approach allows clustering to be applied on some intermediate level of a hierarchy. It is observed that in some cases, clustering at higher level introduces more cohesive clustering, in addition to faster clustering due to fewer elements in each vector. For example, when q=1, it can be seen that the distance between user i and j measured on parent zone is always shorter because of the following inequation:

$$|(C_{i1}+ C_{i2})/M_i - (C_{j1}+C_{j2})/M_j| \leq |C_{i1}/M_i\text{-}C_{j1}/M_j| + |C_{i2}/M_i\text{-}C_{j2}/M_j| \qquad (\mathbf{9})$$

That is, user i and j have more similarities in mobile behaviors on parent zone ($LCA_{1,2}$). To constrain the number of groups, it may be more appropriate to perform clustering on some level other than the leaf level. We can then use the approach described in Section 3 on a per-group basis to determine the replication on the databases at and above the level in the hierarchy. However, since the calling frequencies (C_{ij}) at higher level are determined by the replication of its children zones, we have to first decide the replication of the databases below the level. We propose to adopt Phase-One algorithm in [7] for determining where to replicate for the nodes in the lower levels.

A database j should always replicate user i's profile if the following inequation is satisfied:

$$C_{ij}\, b_l > M_i\, b_u \qquad (\mathbf{10})$$

In other words, if $LCMR_{ij}$ (i.e., C_{ij}/M_i) > b_l/b_u, the decrease of lookup cost is more than the increase of update cost even if the parent of j also replicates user i's profile. In [7], R_{max} is defined to be b_l/b_u, and phase one of their algorithm picks up those databases with $LCMR$s greater than R_{max} in a bottom manner.

Our algorithm also proceeds in two phases. Let L_c be the level in the hierarchy where clustering is to be conducted. In the first phase, for each mobile user, a bottom up traversal in the hierarchy up to level L_c-1 is performed to determine the placement of the profile on a database only if its $LCMR > R_{max}$. Then we cluster the users into groups based on their *LCMR*s. After the first phase, let *K"* be the maximum number of databases that store the profile replicas for each member in a group. In the second phase, we use the dynamic programming approach described in Section 3 to perform RA(*r*, *N*-*K"*) for each group, if *K"* < *N*.

We are still left out with the problems of how to choose the level in the hierarchy for clustering and how clustering should be conducted. As described before, the clustering algorithms in the literature can be classified into two kinds: hierarchical clustering and partitioning. While hierarchical clustering tries to achieve the best clustering by continuously splitting the groups until the result is satisfactory, the partitioning approach aims to accomplish the best clustering for a given cardinality. A straightforward approach is to first use some hierarchical clustering approach on each level in the hierarchy in a bottom up fashion. If the number of groups obtained in a level is not small enough, the next higher level is tried. This procedure continues until the groups are cohesive and the number of groups is reasonably small. Once the level and the ideal number of groups are decided, partitioning is followed.

Up to now one may wonder the usefulness of incorporating clustering because clustering tends to take a long time (approximate time complexity is $O(n^3)$ for agglomerative-nesting algorithm, where *n* is the number of vectors [8]). We argue that the first phase of our approach can be conducted less often, while the second phase is performed more frequently as shown in Fig. 3. The rationale behind such an arrangement is that user's mobile behavior should be stable within a certain long period of time. Such a period determines the interval between the adjacent invocations of the first phase of our algorithm. The second phase of our algorithm, which can be conducted more often, is then used to fine-tune the replication assignment.

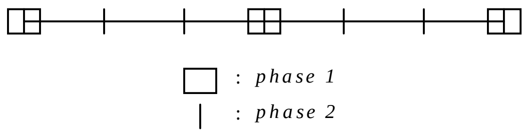

Fig. 3. The two-phases clustering

6 Conclusions

This paper investigates the location replication problem in a hierarchy of databases. With some simplification, we propose a dynamic programming approach for solving this problem. A preliminary experimental result shows that the dynamic programming approach returns better replica assignment in most cases. To further reduce the overhead of storage requirements and execution complexity, we incorporate clustering techniques which group mobile users with similar mobility behavior. More comprehensive performance evaluation is currently under way for thorough comparisons.

References

1. M. J. Berry, and G. Linoff, Data mining techniques: for marketing, sales, and customer support, John Wiley & Sons, 1997.
2. H. Harjono, R. Jain, and S. Mohan, "Analysis and simulation of a cache-based auxiliary user location strategy for PCS," Proc. of Int'l. Conf. on Networks for Personal Communication, 1994.
3. A. K. Jain, and R.C. Dubes, Algorithms for clustering data, Prentice-Hall, 1988.
4. R. Jain, Y-B. Lin, C. Lo, and S. Mohan, "A caching strategy to reduce network impacts of PCS," IEEE Journal on Selected Areas in Communications, 12(8):1434-1444, Oct.1994.
5. R. Jain, and Y-B. Lin, "A auxiliary user location strategy employing forwarding pointers to reduce network impacts of PCS," Wireless Network, 1:197-210, 1995.
6. R. Jain, "Reducing traffic impacts of PCS using hierarchical user location database," Proc. of the IEEE Int. Conf. on Communcations, 1996.
7. J. Jannink, D. Lam, N. Shivakumar, J. Widom, and D.C. Cox, "Efficient and flexible location management techniques for wireless communication systems," Wireless Networks, 3:361-374, 1997.
8. L. Kaufman, and P. J. Rousseeuw, Finding groups in data: an introduction to cluster analysis, John Wiley & Sons, 1990.
9. P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Static and dynamic location management in mobile wireless networks," Journal of Computer Communication, 19(4), March 1996.
10. S. Mohan, and R. Jain, "Two user location strategies for personal communication services," IEEE Personal Communications, 1(1): 42-50, 1994.
11. R. T. Ng, and Jiawei Han, "Efficient and effective clustering methods for spatial data mining," Proc. of Int'l. Conf. on VLDB, 1994
12. E. Pitorua, and I. Fudos, "Tracking mobile users using hierarchical location databases. Technical Report DCS-97-10, Department of C.S., University of Ioannina, 1997.
13. E. Pitoura, and G. Samaras, Data management for mobile computing, Kluwer academic publishers, 1998.
14. G.D. Ramkumar, and A. Swami, "Clustering data without distance functions," IEEE Data Engineering Bulletin, 21(1):9-14, Mar. 1998.
15. N. Shivakumar, and J. Widom, "User profile replication for faster location lookup in mobile environments," Proc. of Int'l. Conf. on Mobile Computing and Networking (Mobicom'95), 161-169, October 1995.
16. J. Z. Wang, "A fully distributed location registration strategy for universal personal communication systems," IEEE Journal on Selected Areas in Communications, 11(6): 850-860, August 1993.