Mobile Data Overlay (MDO): A Data Placement Paradigm for Mobile Applications

Yun Huang
School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697, USA
yunh@ics.uci.edu

Nalini Venkatasubramanian
School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697, USA
nalini@ics.uci.edu

Abstract

This paper addresses the issue of proxy-based data placement for mobile applications. The key idea is to use aggregated information about mobile users and their data request patterns to determine when, how frequently and how much data to be replicated on proxies. Specifically, we introduce a novel representation, Mobile Data Overlay (MDO) that captures aggregated mobile data access patterns. The underlying representation of the MDO utilizes an interval tree-based data structure in supporting efficient spatio-temporal operations on mobile data access information. We develop intelligent MDO reconfiguration (ReC-MDO) techniques that help determine proper granularity of data replication operation (i.e. appropriate segmentation of each data object) by extracting spatio-temporal locality of mobile data access patterns. The ReC-MDO approach effectively balances tradeoffs between replication cost and data access cost in making mobile data placement decisions on proxies. Through extensive experimentation, we demonstrate the superiority of our techniques over several popular data placement strategies.

1 Introduction

Recent advances in wireless networking and device technologies have enabled applications for mobile and wireless users, including but not limited to: video streaming and messaging and video blogging. In this paper, we show how network proxies that connect to mobile hosts via wireless access points or base stations can be effectively used to support data intensive mobile applications, e.g. video streaming. More specifically, we select nearby proxies (that are close to mobile users) and replicate read-only data requested by mobile users on them. The advantages of proxy-based data placement are manifold. Firstly, it reduces the load on the original data server, which maintains original data objects and services a large number of requests. Secondly, since data is available closer to the user, data access delays to mobile users are reduced resulting in better Quality-of-Service (QoS). Thirdly, localized placement of data can help reduce overall network load in the wide area network. Depending on how frequently data placement decisions are made, we can categorize existing data placement strategies into long-term data placement and short-term caching techniques. Our specific focus is on long-term data placement that addresses data needs of mobile users over longer periods of time. Efficient long-term placement of data in a mobile network can significantly reduce on-demand caching and replication overheads.

However, the long-term multiuser data placement problem poses many research challenges. Mobile users move freely. A proxy that is optimal for mobile users at one point in time may be not be optimal throughout the service duration. Capturing and representing the changing relationships between mobile data access patterns and user mobility present a significant challenge. Furthermore, individual user data request patterns are likely to change. Long-term data placement methods should be resilient to the above changes. Additionally, resource allocation also needs to consider heterogeneous and varying resource capacities at a proxy so as to improve its resource utilization.

The key idea of our approach is to apply aggregated data request patterns and mobility behaviors of large number of users for making long term data placement decisions. In the remainder of this paper, we first model the mobile data placement problem in Section 2. In Section 3, we introduce a novel notion called Mobile Data Overlay (MDO), which represents aggregated information of mobile data access patterns. In Section 4, we elaborate on how to use the MDO to make data placement decisions. We evaluate the performance of our MDO-based approach in Section 5 and conclude our work in Section 6.

978-0-7695-3154-0/08 $25.00 © 2008 IEEE
DOI 10.1109/MDM.2008.41

2 The Mobile Data Placement Problem (MDP)

We now model the mobile data placement problem and present our approach. We characterize the problem as the following: every T units of time, the system makes decisions about replicating data objects on proxies. In the mobile data placement, a placement decision between a data object and a proxy may not span the whole T, therefore a single request by a mobile host may be serviced by multiple proxies during its lifetime. The motivation for using multiple proxies arises from the fact that nearby proxies (as a mobile host moves through the network) would significantly reduce network resource consumption as opposed to proxies further away in the network. However, creating multiple replicas across proxies may increase cost of connection setup and replication. According to the data placement problem, without losing generality, the replication cost is associated with the replication operation that stores a specific amount of data on a proxy.

Our model is comprised of four key entities: mobile users, data items requested by mobile users, wireless regions, and network proxies (the proxy set includes the original server that holds content needed by mobile users). Let $U = \{u_i\}$ $(1 \leq i \leq |U|)$ be the group of mobile users ($|U|$ is the total number of mobile users), $D = \{d_q\}$ $(1 \leq q \leq |D|)$ be the set of data items (L_q is the size of data item d_q), $W = \{w_n\}$ $((1 \leq n \leq |W|)$ be the wireless regions that are under considerations, and $P = \{p_j\}$ $(1 \leq j \leq |P|)$ represent the set of proxies and the original server in the network. Each proxy p_j has *K* types of resources (e.g. CPU, storage, and network bandwidth) that will be used when replicating the data and sending the data to the mobile host. For example, $R_{j,k}$ represents the amount of type *k* resource that proxy *j* has, $where\ k \in \{cpu, network, memory\}$. The replication cost $RC_{j,q}(t)$ can be defined as the total storage cost for replicating d_q on the proxy p_j and one time processing cost (cpu or network) to perform the replication at time t [8]. Each user has itinerary information with data access patterns, M_i. It consists of a sequence of tuples, each being defined as $< t_m, w_m, \tau_m, d_m >$, which represents that at time instance t_m the mobile user i requests data item d_m at region w_m for τ_m period of time ($1 \leq$ m $\leq |M_i|$). We let $QoS_{i,k}$ represent mobile user i's QoS requirement on type k resource of proxies. We define $AC_{i,j,q}(t)$ as the data access cost, i.e. network resource consumption of the connection between user *i* and proxy *j* for accessing data item *q*.

Let $f_{j,q}(t)$ represent replication decision on whether to replicate d_q on proxy j at time instance t, and $x_{j,q}(t)$ indicate the time duration for which data d_q will be on proxy *j* after a replication performed at time t. Notice that $\forall t, x_{j,q}(t) \leq T$ (as placement decisions will be made periodically). Let $y_{i,j,q}(t)$ represent if mobile client *i* connects to proxy *j* to access data d_q at time t. Suppose that the data placement decision is made every T units of time (placement period), then the objective function of mobile data placement is to minimize the sum of replication cost and data access cost, as shown in Equation (1), where: if $x_{j,q}(t) > 0$ then $f_{j,q}(t) = 1$, otherwise $f_{j,q}(t) = 0$; and if $i\ accesses\ data\ d\ from\ j\ at\ time\ instance\ t$ then $y_{i,j,q}(t) = 1$, otherwise, $y_{i,j,q}(t) = 0$.

$$
\begin{aligned}
min \quad & \sum_j \sum_q \int_{t=0}^{T} f_{j,q}(t)\ RC_{j,q}(t)\ dt \\
& replication\ cost \\
& + \sum_i \sum_j \sum_{m \in |M_i|} \int_{t_m}^{t_m + \tau_m} y_{i,j,m}(t)\ AC_{i,j,m}(t) dt \\
& data\ access\ cost
\end{aligned}
\tag{1}
$$

Once $x_{j,q}(t)$ is determined, the value of $f_{j,q}(t)$ is decided, but the values of $x_{j,q}(t)$ and $y_{i,j,q}(t)$ are subject to the following conditions:

1. $\sum_j y_{i,j,q}(t) = 1$, indicating that at any time instance exactly one proxy (including the original data server) must satisfy the request of client *i* for data d_q;

2. $y_{i,j,q}(t) \leqslant \begin{cases} 1, if\ exists\ t_1\ and \\ \quad 0 \leqslant t_1 \leqslant t < t_1 + x_{j,q}(t_1) < T \\ 0, \quad otherwise \end{cases}$, indicating that if a mobile user i is assigned to proxy j for data q, the replica of d_q must exist on proxy j;

3. $\sum_q x_{j,q}(t) L_q \leq available\ storage\ of\ j\ at\ time\ t$, indicating that the total size of replicated data items on proxy *j* can not exceed its available capacity;

4. $\sum_i \sum_q y_{i,j,q}(t)\ QoS_{i,k} \leq R_{j,k}$, indicating that each type of resources consumed for accessing data on proxy *j* can not exceed the total available resource.

Given locations of fixed proxies and available resources $R_{j,k}$, we can calculate data replication cost $RC_{j,q}(t)$ and data access cost $AC_{i,j,q}(t)$. Values of $x_{j,q}(t)$ and $y_{i,j,q}(t)$ required to minimize costs defined in Equation (1) correspond to individual mapping/placement decisions, which place required data onto selected proxies to serve mobile requests.

2.1 Hardness of The Problem

Let us consider the dynamic mobile data placement problem defined above as a one-time data placement problem. For example, $x_{j,q}(t)$ once determined does not change for the entire placement period T. Then we can remove t of $x_{j,q}(t)$, $RC_{j,q}(t)$ and $f_{j,q}(t)$, thus:

$$x_{j,q} = \begin{cases} T, if\ d\ is\ replicated\ on\ j\ for\ T \\ 0,\ otherwise \end{cases}$$

and $f_{j,q} = \frac{x_{j,q}}{T}$.

Let us further constrain mobility by assuming that: (1) mobile users are in fixed locations during T; (2) a request for a data item d_q can be served by only one proxy. Thus, we can drop the t from $y_{i,j,v}(t)$ and $AC_{i,j,q}(t)$ as well, such that: $y_{i,j,q} \leq \frac{x_{j,q}}{T}$ according to condition 2). Finally, the objective function defined in Equation (1) can be rewritten as (L_i is data access duration of mobile host i):

$$\begin{aligned} min \quad & \sum_j \sum_q \int_{t=0}^{T} f_{j,q}(t)\ RC_{j,q}(t)\ dt \\ + \quad & \sum_i \sum_j \sum_{m \in M_i} \int_{t_m}^{t_m+\tau_m} y_{i,j,m}(t)\ AC_{i,j,m}(t)\ dt \\ \Rightarrow \quad & \sum_j \sum_q T\ f_{j,q}\ RC_{j,q} \\ + \quad & \sum_i \sum_j \sum_{m \in M_i} \tau_m\ y_{i,j,m}\ AC_{i,j,m} \\ \Rightarrow \quad & \sum_j \sum_q x_{j,q}\ RC_{j,q} \\ + \quad & \sum_i \sum_j L_i\ y_{i,j}\ AC_{i,j} \end{aligned} \tag{2}$$

In this particular case, the problem can be easily interpreted as the ILP presented in [1], which has been proved to be NP-Hard by means of an approximation-preserving reduction from a special case of the uncapacitated facility location problem [3]. Thus our problem is NP-Hard.

2.2 Related Work and Our Approach

Several hueristics have been developed for non-mobile users [1, 8] (e.g. for web data [10]). However approximation solutions are computationally intensive and mobility of users is not a criteria in their design. Decisions on where and when to cache data for mobile applications largely rely on each user's itinerary information. A heuristic [11] is proposed for replicating data objects on mobile hosts (not on proxies) to reduce communication cost of mobile transactions, where placement decisions are made based on correlations of data accessed by one mobile host. Effectiveness of applying user mobility patterns in devising data allocation schemes has been demonstrated by [9], where mining techniques are applied on individual user moving patterns. They are unsuitable for addressing a large number of mobile requests. Hara [7] presented methods for placing replicas on mobile hosts in ad hoc networks. But the assumption is that access frequencies to data items from each mobile host are known and do not change.

Our conjecture is that per-use based optimization is inefficient for long-term data placement that aims at optimizing performance on a global scale. In this light, we pursue the issue of long-term (median-term) and multiple-user data placement for mobile applications. Our idea is based on two observations. First, aggregating data needs of users will enhance data locality in wireless regions over time. Second, selecting the appropriate granularity for object replication will enable better proxy utilization. Therefore, instead of explicitly using individual mobile host's itinerary, we propose to use the aggregated mobile data access information to make data placement decisions.

In this paper, we apply a brokage service, where a broker receives all mobile requests. Once the broker schedules requests to proxies, it converges users' data access patterns with users' mobility information and stores aggregated access and mobility information in a directory service (note that individual user's mobility and data request information is not stored in the directory service explicitly). Periodically, the broker makes placement decisions for the next period. It aims to reduce the total replication and data access cost by: (1) determining suitable granularity (optimal segmentation) of a data item for replication, (2) deciding proper number of replicas for each data item, (3) choosing when to replicate a segment of a data item and (4) selecting proxies for each segment replication. The major contribution of this paper lies in how we tackle the issue of determining proper granularity of data placement by applying spatio-temporal locality of aggregated mobile data accesses, so as to balance the tradeoff between replication cost and data access cost. In this paper, we only consider read-only data.

3 Mobile Data Overlay

In this section, we introduce the notion of a Mobile Data Overlay (MDO) that captures aggregated data request pattern for a large number of mobile users. Specifically, the MDO keeps information on the total number of requests from each wireless region w that are issued for a certain part of data item, e.g. video object o during a period of time, e.g. τ. The MDO is maintained by the directory service of the broker. The rationale for aggregating mobile data access patterns is manifold. Firstly, mobility of users is correlated with space and time [2, 5]. User movement within a given time period is restricted to a region. Consequently, mobile data access patterns in a mobile wireless network are correlated with space and time, specifically for long-term sessions. This is especially true when users move along in well-known trajectories, e.g. freeways. Secondly, aggregated information on the number of users and where they request the data objects can be used to study the correlation among data needs of users over wireless regions, i.e. popularity of data. Furthermore, data segmentation techniques [12] have been proposed as efficient solutions to reduce system cost and improve system throughput. Considering that the Mobile Data Overlay (MDO) maintains aggregated data request patterns at wireless region level over different periods of time, we can further define and refiner

granularity of data objects (e.g. different portions of a data object) kept in the MDO. Thus, the MDO can capture "data mobility" in a finer granularity instead of directly representing "user mobility" or requests information of individual data objects.

We now present the data structure used to represent the MDO and specify operations that efficiently access and update the MDO. The MDO needs to contain both temporal and spatial information. As we try to collect aggregated data access information without identifying individual mobile data access, spatio-temporal data structures used in moving object databases are too complex for our use. Also because spatial information (the partition of wireless regions) usually does not change as dramatically as temporal information (both in terms of granularity and scale), we propose to use an interval tree-based [4] data structure to present the MDO. Let H be the height of the MDO tree. As shown in Fig. 1, we use a BST (Binary Search Tree), where each node represents a time interval $[a, b]$, $a < b$ in a timeline. The leaf nodes partition the time line into "elementary intervals" that are disjoint. The set of leaf nodes are chosen such that they cover the entire timeline. Each internal node of the tree spans the intervals in its child nodes (left and right subtrees). Spatial information (e.g. geographical distribution of concerned wireless regions) is contained within each node of the tree. We use an individual ID number to identify the spatial area of a wireless coverage. Each node in the MDO contains information about requests for data item segments at the different wireless regions in the network.

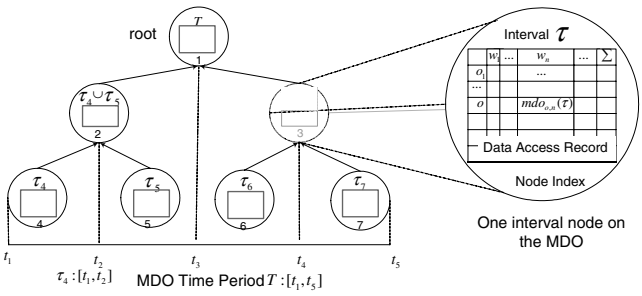


Figure 1. The MDO tree.

Given the above interval-tree based data structure, updating the MDO is to insert individual mobile users' data request pattern information along with their mobility information into corresponding nodes on the MDO tree. This operation will transform one data object request into requests to each data item segments of this data object and update them into the MDO. Information of aggregated request issue time and the user's itinerary information will be used to make data placement decisions for the next MDO period. Thus, it can be collected or computed either when the system schedules a request of a mobile host or after the service finishes. Individual user's mobility prediction is not required for this system. We use a complete binary tree to represent the MDO, and we enumerate the interval nodes of the tree from the top level (root) to the leaf level (root index is 1). If the parent's index is *i*, its left child's index is found to be *2i*, and its right child's index is $2i+1$. The complexity of updating interval node and querying data access information of an interval are $O(1)$.

Based on the notion of the MDO (each item of the MDO's interval node can be denoted as $mdo_{o,w}(\tau)$), the problem defined by Equation (1) in Section 2 can be rewritten as the following Equation (3),

$$\begin{aligned} min \quad & \sum_j \sum_q \int_{t=0}^{T} f_{j,q}(t)\ RC_{j,q}(t)\ dt \\ & + \sum_i \sum_j \sum_{m \in M_i} \int_{t_m}^{t_m+\tau_m} y_{i,j,q}(t)\ AC_{i,j,q}(t) dt \\ \Rightarrow \quad & \sum_j \sum_o \sum_\tau x_{j,o}(\tau)\ RC_{j,o}(\tau) \\ & + \sum_n \sum_j \sum_o \sum_\tau y_{n,j,o}(\tau)\ AC_{n,j,o}(\tau) \end{aligned} \tag{3}$$

where, $x_{j,o}(\tau)$ denotes if data *o* on the MDO will be replicated onto proxy *j* at the beginning of time interval τ and will be kept on proxy *j* for time period of τ. $RC_{j,o}(\tau)$ depicts the replication cost of object *o* on proxy *j* during τ. $y_{n,j,o}(\tau)$ represents the number of requests served by proxy *j* for object o from wireless region *n* during time interval τ. $AC_{n,j,o}(\tau)$ is the data access cost per request between wireless region *n* and proxy *j* when the requests for object o occur during time τ. If $mdo_{o,n}(\tau)$ is the predicted total number of requests for object o from wireless region *n* during time τ, $y_{n,j,o}(\tau)$ will not be larger than $mdo_{o,n}(\tau)$.

4 Mobile Data Placement by Reconfigurable MDO (ReC-MDO)

We now develop a mobile data placement algorithm that addresses the problem, defined in Equation (3). Specifically, we determine if proxy j replicates object o (represented by $x_{j,o}(\tau)$) and the number of requests served by proxy j for object *o* from wireless region n during time period τ (represented by $y_{n,j,o}(\tau)$).

4.1 MDO Reconfiguration

We have two major concerns when using MDO data records to determine which proxies should replicate which video segmentation. The first concern is that, "raw" data collected in nodes of smaller intervals (at lower level of the MDO tree) is at finer temporal granularity, so it is more accurate than that of larger intervals (at relatively higher level of the MDO tree). Intuitively, to reduce the data access cost, we can directly search available proxies and make replications for requested video objects. However this may lead to

the result that the same data segment will be replicated on two different proxies for two contiguous intervals, which increases the number of data replication operations and increases data replication cost. By exploiting temporal correlations of mobile data access information, we may be able to merge these two data access records into one and store it at higher level of the tree. Such that when we make placement decisions at the higher level, there is a higher chance to find one proxy to replicate this data segment for two contiguous intervals. In this case, we can reduce replication cost. However decisions made at higher level nodes may be unable to exploit fine grained access information held at lower nodes in the tree, which are originally designed to help reduce data access cost. The second concern is that, when there are data access records for two contiguous video segments belonging to the same data object in one interval node, if we try to select proxies for them individually, we may also find two different proxies to replicate them separately (which may increase the replication cost). If we merge them together as a bigger segment, we will get higher chance to place them on the same proxy. There are other advantages to fusing smaller data segments into larger segments. Firstly, the replication and access costs for the segments are reduced since connection establishment and teardown will be performed fewer times. Furthermore, such continuity in common applies to dealing with continuous media. Reducing proxy switches helps improve QoS.

The above two concerns raise about a tradeoff between replication cost and data access cost. Our challenges lie in making effective data placement for "longer" intervals (thereby reducing replication cost) without losing useful information contained in the raw data held at "shorter" intervals nodes. We solve the above problem by reconfiguring the MDO, called ReC-MDO. The basic strategy in the ReC-MDO is to push "similar" data object together and higher up in the interval tree while remaining cognizant of data access cost incurred as data migrates up the tree. According to the above two concerns, we will address the following two similarities between mobile data access patterns. First, if data access patterns for a single object are "similar" over contiguous time intervals, the object can be moved to a higher level node. We refer to this as object similarity (*ObjSim*). Second, if "contiguous" segments of a single data item that are accessed in contiguous intervals have "similar" data access patterns, they can be merged into a single segment higher up in the interval tree. We refer to this as continuity similarity(*ConSim*).

In this paper, we apply standard deviation to study *ObjSim* between $\overrightarrow{mdo_o(\tau_1)}$ and $\overrightarrow{mdo_o(\tau_2)}$. The *ObjSim* is defined by Equation (4), which applies definitions defined by Equation (5) and (6). Note that in order to avoid being divided by zero, we add 1 to each element of $\overrightarrow{mdo_o(\tau_2)}$ in Equation(5). We quantify *ConSim* using both spatial and temporal factors by calculating Euclidean distance between two vectors: $\overrightarrow{mdo_{o_1}(\tau_1)}$ and $\overrightarrow{mdo_{o_2}(\tau_2)}$, as defined by Equation (7). Notice that smaller values of *ObjSim* and *ConSim* indicate higher similarities. The *ObjSim* and *ConSim* are used to enable object-proxy mapping decisions, i.e. determining $x_{j,o}(\tau)$ values by reconfiguring the MDO.

$$\begin{aligned} ObjSim_{o,\tau_1,\tau_2} &= \sigma(mdo, \tau_1, \tau_2, o) \\ &= \sqrt{\frac{1}{|w|}\sum_{n=1}^{|w|}(diff_{o,n,\tau_1,\tau_2} - ave_{o,\tau_1,\tau_2})^2} \end{aligned} \tag{4}$$

$$\begin{aligned} \overrightarrow{diff_{o,\tau_1,\tau_2}} &\stackrel{def}{=} \frac{\overrightarrow{mdo_o(\tau_1)}}{\overrightarrow{mdo_o(\tau_2)}} \\ &\stackrel{def}{=} \begin{bmatrix} diff_{o,1,\tau_1,\tau_2} \\ \vdots \\ diff_{o,|W|,\tau_1,\tau_2} \end{bmatrix} = \begin{bmatrix} \frac{mdo_{o,1}(\tau_1)}{mdo_{o,1}(\tau_2)+1} \\ \vdots \\ \frac{mdo_{o,n}(\tau_1)}{mdo_{o,n}(\tau_2)+1} \end{bmatrix} \end{aligned} \tag{5}$$

$$ave_{o,\tau_1,\tau_2} \stackrel{def}{=} \frac{1}{|W|}\sum_{n=1}^{|W|} diff_{o,n,\tau_1,\tau_2} \tag{6}$$

$$\begin{aligned} ConSim_{o_1,o_2,\tau_1,\tau_2} &= distance(mdo, \tau_1, \tau_2, o_1, o_2) \\ &= \sqrt{\sum_{n=1}^{|W|}[mdo_{o_1,n}(\tau_1) - mdo_{o_2,n}(\tau_2)]^2} \end{aligned} \tag{7}$$

The overall flow of the Reconfigurable MDO (ReC-MDO) data placement algorithm (illustrated in Fig. 2) is as follows.

PROCEDURE: Mobile_Data_Placement_by_Reorganizing_MDO
1. FORALL level $l = mdo.height$ to 2
2. FORALL node $p = 2^{l-2}$ to $p = 2^{l-1} - 1$
3. $lC \leftarrow 2p$; // left child index
4. $rC \leftarrow 2p + 1$; // right child index
5. IF $mdo.lC! = NULL$
6. FORALL data objects $lC.o$ in $queue$
7 calculate ObjSim(lC, rC, o), ConSim(lC, rC, o, o')
8. IF similar AND resource is sufficient
9. reorganize_MDO($p, lC.o, rC$)
10. ELSE
11. Select_Proxies_for_An_Object($\overrightarrow{mdo_{o_{lC}}(\tau_{lC})}$)
12. FORALL data objects o_{rC} in rC
13. Select_Proxies_for_An_Object($\overrightarrow{mdo_{o_{rC}}(\tau_{rC})}$)
14. FORALL data objects o_{root} in $root$
15. Select_Proxies_for_An_Object($\overrightarrow{mdo_{o_{root}(\tau_{root})}}$)
END PROCEDURE

Figure 2. Data placement of ReC-MDO.

We initiate the object proxy mapping process from the first leaf node. We start comparing data access similarity between data objects within each pair of leaf nodes by calculating *ObjSims* and *ConSims* between all object entries in a left child node and their relevant object entries in right child node. If the data entries of a left child and a right child are similar, and there are proxy resources available for the

new merged data object, we push the object and its correlated data object at the sibling node together up to the tree, as illustrated in Fig. 3. Otherwise, this object remains at its current level in the interval tree, and we run a separate algorithm, which will be presented in Section 4.2, to select proxies for this object. Once we finish mapping the objects in the left child node, we continue mapping the objects onto proxies in the right child node. The above comparing and merging procedure will be executed iteratively on each pair of sibling nodes and their parent nodes until we reach the root of the MDO tree. The ReC-MDO process automatically determines proper segment size by evaluating *ConSim* and inherently influences the number of replicas needed for each object, as well as the number of replication operations by studying *ObjSim*.

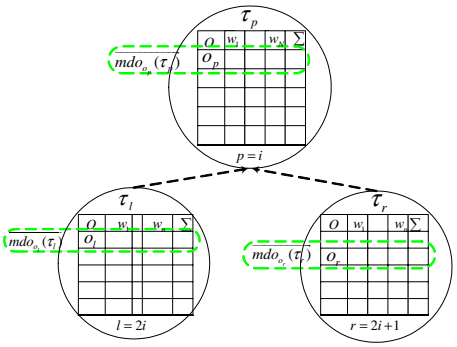


Figure 3. Reconfigure the MDO.

4.2 Determining Object-Proxy and Service-Proxy Mapping

We now address the replication of each individual object. Specifically, we determine the selection of proxies on which to place replicas, i.e. (1) $x_{j,o}(\tau)$, mapping of objects to proxies; if object o is replicated on proxy j for interval τ, then set the $x_{j,o}(\tau) = 1$; (2) $y_{w,j,o}(\tau)$, potential mapping of arriving service requests to replicated objects on proxies. In order to select proxies for an object, we firstly populate a cost table, as shown in Fig. 4, which provides data access cost between wireless regions and proxies. We then evaluate data access cost of selecting proxy j to replicate object o for time period τ based on the following equation:

$$\begin{aligned} AC_{o,j}(\tau) &= \overrightarrow{mdo_o(\tau)} \cdot \overrightarrow{AC_{j,o}(\tau)} \\ &= \sum_{n=1}^{|W|} (mdo_{o,n}(\tau)\ AC_{n,j,o}(\tau)). \end{aligned} \tag{8}$$

We then provide a generalized algorithm, as shown in Fig. 5, to select nearby proxies for a segment object of a data item. A proper selection must satisfy the following conditions: (1) the proxy is available for the requested period of time; (2) proxies have sufficient resource for replicating the

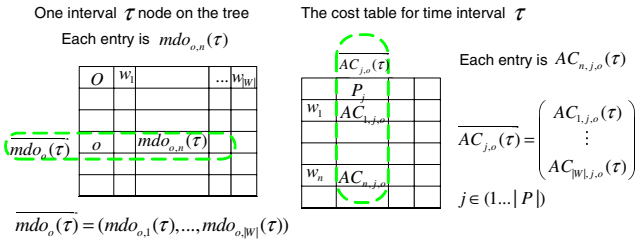


Figure 4. Cost matrix.

object; (3) the selection of the proxy produces a minimal cost mapping where cost is defined as Equation (8). The selected proxy may have insufficient resources to accommodate all requests. We can either greedily select proxies in order of increasing access cost, or we try to balance proxy loads by selecting proxies in order of decreasing available resources to fill as many predicted requests as possible.

PROCEDURE: OBJECT AND PROXY MAPPING

1. $candidateProxies \leftarrow$ available proxies resource
   // consider proxy time availability for τ and
   // sufficiency for providing predicted services of object o
2. FORALL proxy j in $candidateProxies$
3. calculate $AC_{o,j}(\tau)$ defined in Equation (8)
4. $selectedProxies \leftarrow$ selected proxies
   // that have sufficient resources to provide predicted
   // services with minimum data access cost
5. FORALL proxy $j \in selectedProxies$
6. $x_{j,o}(\tau) \leftarrow 1$
7. FORALL wireless region $w_n \in \overrightarrow{mdo_o(\tau)}$
8. $y_{n,j,o}(\tau) =$ the number of predicted requests that proxy j can supports
9. reduce predicted available resources of interval τ

END PROCEDURE

Figure 5. Select proxies for an object

At the end of every MDO period, the ReC-MDO placement procedure is executed to determine mappings between data objects and proxies for the next MDO period. The expectation here is that requests that arrive in the next period will be mapped to the replicated segments on proxies. We enforce the request-proxy mappings by suggesting proxies that handle requests from wireless regions through appropriate initialized value of $y_{n,j,o}(\tau)$. This inherently provides a substrate for mapping incoming requests to suitable proxies. Then, during the next period, when a new mobile request arrives at t_m requesting data d_m from region w_m, the broker will select a proxy j that has the requested data, i.e. $x_{j,d_m}(\tau_m)$ equals to 1. The selected proxy p_j must have sufficient network resources to enable mobile host's data access, and the data access cost $AC_{w_m,j}(\tau_m)$ incurred by this service connection hopefully be the smallest among all available proxies that can provide the data to the mobile host. In fact, under certain conditions, balanced scheduling, i.e. scheduling this request to a "powerful" proxy may not lead to the minimal access cost, but is beneficial to the whole system in reducing total cost of data accesses. Once we decide the service mapping between the mobile host at a

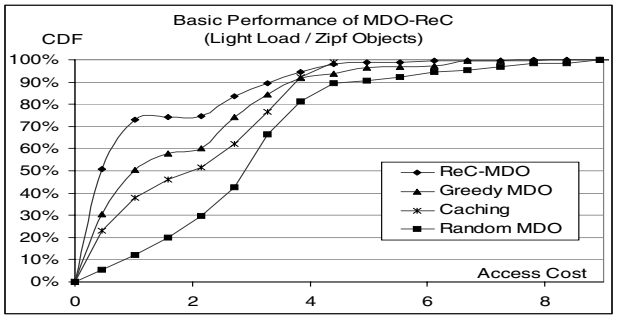
region w_m during interval τ_m, the data access record of the current MDO, e.g. $mdo_{d_m,n_m}(\tau_m)$ is increased by 1, and this information will be leveraged for the next mobile data placement process.

5 Performance Evaluation

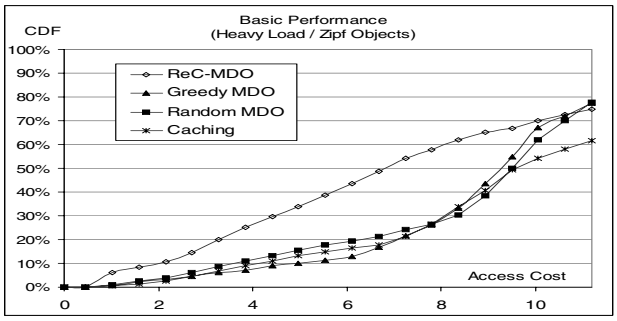
In this section, we present and analyze our simulation results. We model a cellular network system with 100 cells, 25 evenly distributed proxies and one data server that keeps the original copies of all data objects. Mobility of individual user is characterized by applying the incremental mobility model [6], where mobile users can move at walking or driving speed in a closed coverage area. We model each user's data requests following the zipfian distribution [8]. This request pattern determines the number of requests that will be issued to different data objects by users. Each user sends video streaming request, and the duration of each video varies from 30 minutes to 4 hours (e.g.two soccer games). The smallest size of a segment unit is 10-minute, thus a 30-min video has at most 3 segments. We performed a set of experiments to evaluate our solutions under various configurations of the simulated system. The basic configurations are as follows. Each proxy has 100 GB storage and 100Mbps network bandwidth. Each data access from the mobile user will consume network transmission bandwidth ranging from 500 kbps to 1.3 Mbps.

The basic performance of ReC-MDO is compared with that of three selected data placement strategies described below. As shown in [8], popularity-based greedy data placement has been regarded as an efficient solution. The greedy MDO policy greedily selects more powerful proxies (with more available network bandwidth) to allocate replicas for popular objects. The random MDO policy randomly selects proxies for popular replicas. The on-demand caching strategy executes a LFU cache replacement and it tries to select the proxy that is closest to a mobile user along his trajectory.

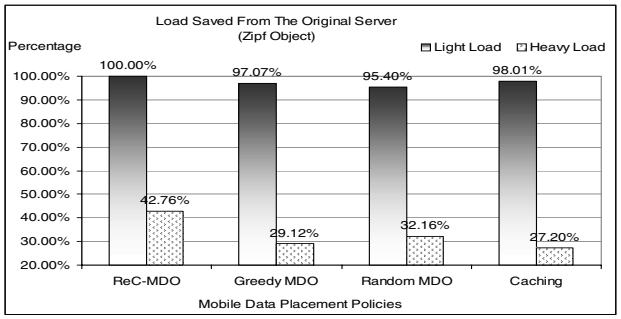
Fig. 6 shows how these policies perform when the system is storage constrained [8], i.e. the total number of replicas is limited. To make a fair comparison, we compare data access cost by restricting each policy to have the same data replication cost. Comparison is made under the same system load and data request pattern. In Fig. 6 (1-3), we present Cumulative Distribution Function (CDF) values of user data access cost over light load and heavy load. The ReC-MDO results are shown to have lower data access cost than the other three policies under different conditions for the majority of requests. For example, in Fig. 6 (1), more than 70% MDO-ReC requests have data access costs lower than 1 cost unit, however only 50% Greedy MDO requests have access costs lower than 1 cost unit. Besides, the other three policies (especially the caching policy) are more sensitive to data request patterns and system workload. As shown in



(1)



(2)



(3)

Figure 6. Basic Performance of ReC-MDO.

Fig. 6 (3), the ReC-MDO also prevails in saving more or equivalent load from the original server than other policies, which indicates an effective use of proxy storage by properly segmenting the data objects.

Fig. 7 illustrates how MDO-based approach performs when different amounts of unexpected requests rush into the system. We restrict the same online replication cost (i.e. online replication cost) for all four policies. Because ReC-MDO is designed to withstand a certain degree of expected request arrivals. As shown in Fig. 7, when heavier unexpected load arrives, e.g. 8 times load increase, the ReC-MDO (Adaptation) exhibits superior performance by executing adaptations with MDO-based approach. This is because ReC-MDO adaptation applies previous MDOs' data records, so that it can make online replication decisions by predicting (e.g. time-series based techniques) the surge in data requests where they are likely to move. Our results

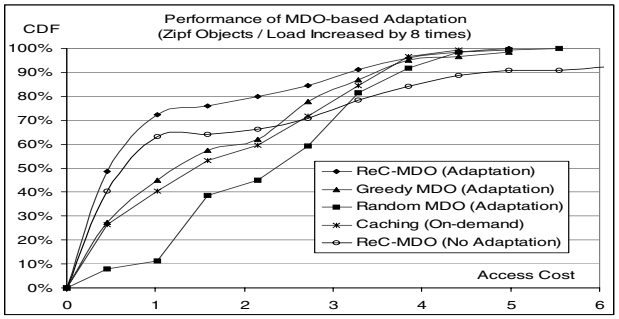
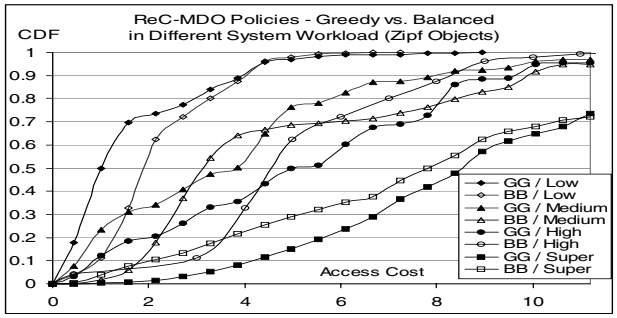
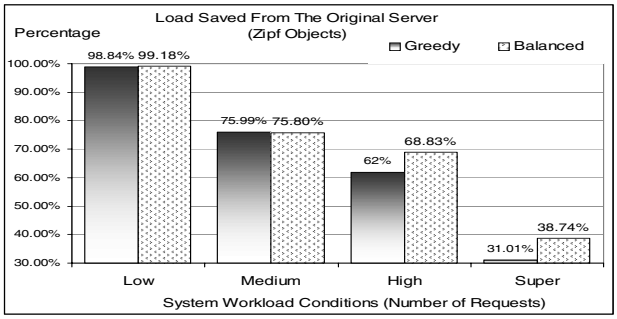


Figure 7. Dynamic Adaptation of ReC-MDO.

also show that when there is less unexpected workload, even without online adaptation, the ReC-MDO can tolerate system load dynamics better than other policies that have additional online adaptation strategies.



(1)



(2)

Figure 8. Studies on ReC-MDO policies.

As discussed in Section 4.2, both greedy and balanced strategies can be used in data placement and scheduling. We evaluated the performance of GG (greedy placement and greedy scheduling), GB (greedy placement and balanced scheduling), BG (balanced placement and greedy scheduling) and BB (balanced placement and greedy scheduling) policies. We find that the placement policy seems to be a differential factor to the final results, i.e. GG and GB have similar results (so do BG and BB). We compare these policies under low, medium, high and super heavy system workloads, as shown in Fig. 8. In brief, we can conclude that load balanced can reduce more workload from the original server, and increases resource utilization of proxies. This advantage is magnified as system workload increases.

6 Concluding Remarks

Based on the intuition that long-term data placement can benefit from the knowledge of aggregated data-access patterns of mobile users, we designed an efficient and flexible data placement strategy to address a set of design tradeoffs. In this paper, we introduced a novel abstraction, the *Mobile Data Overlay(MDO)* that captures aggregated data access patterns of mobile users. We presented how to use intelligent MDO reconfiguration techniques in making efficient proxy data placement decisions. Our simulation results show that the ReC-MDO mobile data placement policy can effectively balance tradeoff between data replication cost and data access cost.

References

[1] I. D. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. In *12th ACM-SIAM SODA*, 2001.

[2] A. Balachandran, G. Voelker, P. Bahl, and P. Rangan. Characterizing user behavior and network performance in a public wireless lan, 2002.

[3] F. A. Chudak. Improved approximation algorithms for uncapacitated facility location. *Lecture Notes in Computer Science*, 1412:180+, 1998.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[5] A. Datta, K. Dutta, H. M. Thomas, D. E. VanderMeer, Suresha, and K. Ramamritham. Analysis of user behavior and traffic pattern in a large-scale 802.11a/b network. In *1st workshop on Wireless Network Measurements*, 2005.

[6] Z. Haas. A new routing protocol for the reconfigurable wireless networks, 1997.

[7] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *INFOCOM*, 2001.

[8] M. Karlsson and C. Karamanolis. Choosing replica placement heuristics for wide-area systems. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, 2004.

[9] W.-C. Peng and M.-S. Chen. Mining user moving patterns for personal data allocation in a mobile computing system. In *ICPP*, 2000.

[10] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *INFOCOM*, 2001.

[11] M. Tu, P. Li, L. Xiao, I.-L. Yen, and F. B. Bastani. Replica placement algorithms for mobile transaction systems. IEEE Transactions on Knowledge and Data Engineering, 2006.

[12] K.-L. Wu, P. S. Yu, and J. L. Wolf. Segment-based proxy caching of multimedia streams. In *WWW '01*, 2001.