An Active Replication Scheme for Mobile Data Management *

Shiow-yang Wu Yu-tse Chang
Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien, Taiwan, R.O.C.
E-mail: showyang@csie.ndhu.edu.tw

Abstract

A replication scheme determines the number and location of replicas in a distributed system. Traditional static replication schemes do not perform well in mobile environment since the assumptions of fixed hosts and relatively static access patterns no longer hold. For effective data management in mobile environment, we propose a dynamic replication scheme which employs user profiles for recording users' mobility schedules, access behavior and read/write patterns, and actively reconfigures the replicas to adjust to the changes in user locations and system status. Simulation results demonstrate that the scheme can accurately predict the data requirement to facilitate effective replication, reduce response time, and increase data availability.

1. Introduction

Data replication [5] is often employed to improve the availability and effectiveness of information services in distributed systems. A *replication scheme* determines the number (*replication level*) and location (*replication placement*) of replicas in a distributed system. Traditional replication schemes [3, 5, 7] are static in the sense that the number and placement of replicas are predetermined and fixed. Manual re-calculation of the access cost and redistribution of replicas are necessary to reflect new access patterns. This is acceptable in traditional distributed environment since the hosts are installed at fixed locations and the access patterns are relatively static. In mobile environment, however, static replication schemes do not perform well because the assumptions about fixed hosts and static access patterns no longer hold [2, 4]. Dynamic replication schemes [1, 13], on the other hand, try to overcome the problem by continuously maintaining statistics about access patterns and system workload so as to dynamically recalculate access cost and reconfigure the replication structure to adjust to the changes in access patterns. This is particularly desirable for mobile computing environment [6, 10, 11]. Even with dynamic replication, these schemes are essentially *passive* in the sense that the actual reconfiguration can only happen **after** the changes in access patterns have been taken place for a period of time and reflected on the access statistics.

We improve upon existing dynamic replication algorithms and propose an active replication scheme which employs user profiles for recording mobile users' mobility schedules, access behavior and read/write patterns, and actively reconfigures the replicas to adjust to the changes in user behavior and system status. Our scheme is unique in several respects: (1) We maintain detail access statistics of each individual user in his/her profile such that the replication decision can be made to tailor the movement and data requirement of each user. (2) We allow a user to provide his/her daily schedule which is used to derive the user's mobility pattern and data requirement. This enables the system to provide a certain degree of predictive replication which allocates object replicas **before** the actual access. (3) We devise the concept of *open objects* to represent a user's current and near future data requirement. This leads to a more precise and responsive cost model to reflect the changes in access patterns. (4) We allow the declaration of emergency events and objects which are unconditionally replicated. This is targeting safety and time critical application domains.

For performance evaluation, we built a simulation environment and compared five representative schemes: (1) no replication; (2) static replication; (3) dynamic data allocation [14]; (4) adaptive data replication [15]; and (5) the proposed active replication. Our scheme can accurately predict the data require-

*This work was supported in part by National Science Council under project number NSC 87-2213-E-259-004.

ment to facilitate effective replication, reduce response time, and increase data availability.

The rest of the paper is organized as follows. Section 2 provides a survey of related issues and research work. Section 3 presents our framework and system model. Section 4 introduces the active replication algorithms. In Section 5, we discuss our simulation method and the results of performance evaluation and comparison. Section 6 concludes the paper.

2. Related Work

The access cost of most dynamic replication schemes are calculated based on the accumulated read/write statistics and the chosen consistency control protocol [6, 11, 15]. Consider the case of deciding whether to replicate an object O on a site. Assume that R is the average number of local read requests per time unit to O on that site, and W is the average number of write requests to O made by all users in the entire system. Let α be the cost that can be **saved** if the object were read locally instead of requested from a remote site, and β be the **additional** cost that must be spent for maintaining a replica of O. Obviously, during a time unit, $\alpha \times R$ is the total cost that can be saved if a replica of O is allocated on the site, and $\beta \times W$ is the total cost that must be spent to maintain this replica. Based on this cost model, we can determine that if $\alpha \times R \geq \beta \times W$ then the allocation of a replica of O on the site is *judicious*. On the other hand, if $\alpha \times R < \beta \times W$ then it is not cost worthy to do so.

Some other design dimensions have also been considered, such as the lower and/or upper limit on the number of replicas of an object [8], as well as the capacity of a site [11]. To set the limit on the number of replicas is to ensure certain level of availability with bounded overhead. Another interesting design is to require that all replicas of an object reside on neighboring sites [15]. This has the advantages of reducing replica allocation and consistency maintenance cost. For objects with strong interdependency, it is sometimes better to replicate the entire group of related objects at the same time. This type of semantic information is used in [9] to improve the efficiency of replication. It is also possible to formulate the replication problem from an economy point of view, i.e. to consider replication as a trade behavior. The servers are the trading parties and the goods for trade are the data objects. Each server can determine whether to buy or sell object replicas, or decide whether to maintain, keep or discard the replicas it has. Such decisions are made based on the data requirement of the local users as well as the maintenance cost to keep the replicas up to date. This approach is taken by the replication subsystem of Mariposa [12].

To summarize, a dynamic replication scheme must determine, among other things: (1) a consistency control protocol; (2) a cost model for estimating the access and replica maintenance cost; (3) the replication structure; and (4) the replication control algorithm(s). The scheme we proposed employs the primary copy model and the ROWA(read-one-write-all) protocol for consistency control, and an improved cost model upon existing dynamic replication schemes for cost estimation. We do not place any limit on the level or placement of object replicas.

3. System Model

The active replication scheme was designed to work in a wired/wireless LAN environment. Servers connected by traditional fixed network are the sites for allocating and maintaining object replicas. Servers that are capable of providing wireless communication services are called *mobile support stations* (MSS). The radio coverage of a MMS is called a *cell*. Any computational device which is carried with a mobile user and is capable of wireless communication is called a *mobile host*. Each user has a *profile* to keep track of the user's current location and related information. A *hand-off* process is taken between two adjacent MSSs to ensure seamless transition when users move from cell to cell. This also implies that a MSS can detect any entrance and exit of users into and out of its cell.

To simplify our algorithm design, we adopt the popular primary copy model and ROWA protocol for replica consistency control. In this model, we select, among an object O and all its replicas (named *O-scheme*), one as the primary copy ($P(O)$). The site that maintains $P(O)$ must have all information about the replicas of O, including the number and locations. All sites that keep a replica of O must know where $P(O)$ resides. All reads to O can be satisfied by the nearest copy of O; while all writes to O must be sent to $P(O)$ first and then propagate to all sites in the *O-scheme* for replica update. The cost of replica update is proportional to the size of the *O-scheme*. We note that more deliberate models and consistency protocols can also be used with our framework. The detail of adopting other models or protocols, however, is not within the scope of this paper.

4. The Active Replication Scheme

Our method improves upon existing dynamic replication schemes with special design toward mobile environment. We employ user profiles to record users'

read/write patterns so that our algorithms can be tailored to satisfy as closely as possible each individual user's information requirement. For predicting future access pattern, we offer the opportunity for a user to specify daily schedule as hints of the user's mobility pattern and data requirement. This is desirable in practice since mobile users do not move at random. They often come to a location at a predetermined time with a specific purpose in mind. Also the work a user is currently engaged in has a strong relationship with the data required. This is why a user's daily schedule can be served as valuable hints for predicting the future. However, a user may not follow schedule strictly. In such case, we resort to past statistics for making replication decision. In using the read/write histories, we propose the concept of *open objects* for better cost estimation. We selectively adopt the read/write statistics of only those objects that are currently in use or likely to be used in the near future. The following sections discuss detail of the active replication algorithms.

4.1 System Information

User Profiles

User profiles are the places for declaring schedules and maintaining per-user access statistics. A schedule is a declaration of daily activities. Each schedule entry declares the time, location, and activity a user plans to do with optional data objects requirement. A mapping is performed to determine the default data requirement of a schedule entry by considering, for example, characteristics of the user, location, and activity. The default requirement is unioned with the explicitly requested data objects to form the information requirement. Figure 1 is an example schedule in a hospital environment. An asterisk signifies an emergency object which are unconditionally replicated. Schedules are explicit hints about users' mobility and access patterns. They have been used regularly to record our daily activities. Where we are and what we do have a strong relationship with what we need for doing our job. Our method simply takes advantages of existing practice for improving information services. A user who appears at the right place and time is called "a user *on schedule*". Otherwise, he/she is called "a user *off schedule*".

The read/write histories are a user's access statistics on data objects in the system. Since a user accesses only part of the data objects at a time and location, we capture this characteristic by defining the concept of *open objects* (in a similar sense as open files) to be the set of objects accessed since entering the current cell and, for a user on schedule, the set of data objects declared explicitly or implicitly in the current schedule

Schedule of Dr. Taylor		
Time	Location	Activity & Objects
00:00 - 01:50	OR1	Operation32
10:30 - 12:00	PD1	Outpatient: ER3906*
14:00 - 16:00	MD3	Outpatient: ER3906*
16:10 - 19:00	OR5	Operation36
21:00 - 21:50	PL1	Project24: PR2409
P.S.		
MD: Medical Dep.		DR: Delivery Room
PD: Pediatrics Dep.		OR: Operation Room
PL: Pathology		ER: Epidemic Report
PR: Pediatric Report		* : Emergency Object

Figure 1. An example schedule.

entry. Open objects represents current and near future data access range. The information is used in calculating the access cost when making replication decision.

Replication Server

A replication server must maintain information about local users and the object replicas, handle read requests and replica update messages. The information about local users includes user profiles and each user's on/off schedule status. For each data object, a server maintains an access record called *local_open_read* which is the sum of all read histories of local users having this object opened. This is used for representing the read pattern of the current cell toward the object. Since the past read histories of users who no longer have this object opened are not counted, our algorithm is more precise and responsive to the changes in access pattern than algorithms that simply accumulate all the past read requests to the object.

For a server that maintains a primary copy of O, a record of the current *O-scheme* must be kept up to date. A global access record called *global_open_write* is used to reflect the current write pattern to object O in the entire system. This is the sum of the write histories of all users who have the object opened.

Emergency Events and Objects

We allow the declaration of *emergency events* and *emergency objects* for must-have objects that require fast access. An emergency event is any situation that demands quick response. In many application domains, such events can be identified in advanced. An emergency event usually has a set of default information requirement that can also be identified beforehand. Whenever such event occurs, the system unconditionally replicates all emergency objects associated with

that event. A user can also declare emergency objects in his/her schedule. To provide emergency services, a server must maintain a *local_emergency_counter* for each object declared as emergent to count the number of emergency events or claimed users to that object. The object is replicated whenever the counter is greater than zero.

System Events

Our algorithms are activated upon the occurrence of certain events. The system events that trigger the execution of the replication control code include the READ event, the WRITE event, the UPDATE event (a server receives a consistency update message), the ENTER/EXIT event (a user enters/leaves a cell), the TIME-CHECK event (the system performs a periodic check for replica management), and the EMERGENCY event (the occurrence of the predefined emergency event). On each event occurrence, a corresponding part of the replication control code is triggered.

4.2 Cost Model

Our cost model follows the basic idea of comparing cost saving of allocating a replica with that of replica maintenance cost (Section 2). The access cost is calculated based on network transmission cost. A local access does not incur any transmission cost while the cost of a remote access is counted as the network distance between the current site and the nearest site with a replica of the desired object. If the distance is d, then the cost per access that can be saved from allocating a replica in the current site is d. Similarly, the extra cost of maintaining the replica on each update is also d since the message is propagated from the nearest site in the *O-scheme*. If a replica of O is allocated in the current site, then the total saving per time unit is $d \times local\_open\_read$, while the extra replica maintenance cost is $d \times global\_open\_write$. Naturally, when $d \times local\_open\_read \geq d \times global\_open\_write$ (or equivalently, , $local\_open\_read \geq global\_open\_write$) it is beneficial to have a replica in the current site. Otherwise it is not cost worthy to do so.

4.3 The Replication Algorithms

For ease of presentation, we define the following symbols:

- $O_1, \ldots, O_k$ are the data objects in the system.
- $B_1, \ldots, B_n$ are the base stations in the system.

READ Algorithm

When (the event U_i reads O_j occurs at B_k)
{
  If (O_j is not an open object of U_i)
  {
    Add O_j to the list of open objects of U_i;
    $local\_open\_read_{kj}$ += $read_{ij}$;
    $global\_open\_write_j$ += $write_{ij}$;
  }

  $read_{ij}$ += 1;
  $local\_open\_read_{kj}$ += 1;

  If (no local replica of O_j exist) **and**
  ($local\_open\_read_{kj} \geq global\_open\_write_j$)
    Allocate a replica of O_j and inform $P(O_j)$;
}

Figure 2. The READ Algorithm.

- $C_1, \ldots, C_n$ are the corresponding cells.
- $U_1, \ldots, U_m$ are the mobile users in the system.
- $P_1, \ldots, P_m$ are the user profiles.
- $S_1, \ldots, S_m$ are the user schedules.

We use $local\_open\_read_{kj}$ to denote the local open read record of B_k on object O_j. Similarly, $global\_open\_write_j$ is the global open write record of the object O_j maintained in $P(O_j)$. The local emergency counter of B_k on object O_j is represented by $local\_emergency\_counter_{kj}$. We also use $read_{ij}$ and $write_{ij}$ to denote the read and write histories of U_i on object O_j. Both of these records are maintained in P_i.

Each server runs an event detector to detect the occurrences of system events described in Section 4.1. Based on the event type, the corresponding replication control module is triggered and executed. There are eight modules for handling different types of situations. The algorithms used in each module are presented in Figure 2 through 9.

5. Simulation and Comparison

We have developed a mobile information system simulation environment which allows us to model and experiment on various network configurations, user mobility and access patterns. We compare the performance of five representative replication schemes:(1) No replication, (2) static replication, (3) dynamic data allocation [14], (4) adaptive data replication [15], and our

WRITE Algorithm

When (the event U_i writes O_j occurs at B_k)
{
  If (O_j is not an open object of U_i)
  {
    Add O_j to the list of open objects of U_i;
    $local\_open\_read_{kj}$ += $read_{ij}$;
    $global\_open\_write_j$ += $write_{ij}$;
  }

  $write_{ij}$ += 1;
  $global\_open\_write_j$ += 1;
}

Figure 3. The WRITE Algorithm.

UPDATE Algorithm

When (an update to a replica of O_j occurs at B_k)
{
  If ($local\_emergency\_counter_{kj} = 0$) **and**
  ($local\_open\_read_{kj} < global\_open\_write_j$)
    Discard the replica of O_j and inform $P(O_j)$;
}

Figure 4. The UPDATE Algorithm.

ENTER Algorithm

When (the event that U_i enters C_k occurs)
{
  If (U_i is on schedule)
  {
    For all O_j required in S_i
    {
      Add O_j to the list of open objects of U_i;
      $local\_open\_read_{kj}$ += $read_{ij}$;
      $global\_open\_write_j$ += $write_{ij}$;

      If (O_j is an emergency object)
        $local\_emergency\_counter_{kj}$ += 1;

      If (no local replica of O_j exist) **and**
      (($local\_open\_read_{kj} \geq global\_open\_write_j$) **or**
      ($local\_emergency\_counter_{kj} > 0$))
        Allocate a replica of O_j and inform $P(O_j)$;
    }
  }
}

Figure 5. The ENTER Algorithm.

EXIT Algorithm

When (the event that U_i exits C_k occurs)
{
  For all open objects O_j of U_i
  {
    Remove O_j from the open objects of U_i ;
    $local\_open\_read_{kj}$ -= $read_{ij}$;
    $global\_open\_write_j$ -= $write_{ij}$;

    If (O_j is an emergency object requested)
      $local\_emergency\_counter_{kj}$ -= 1;

    If (O_j has a replica in the site) **and**
    ($local\_emergency\_counter_{kj} = 0$) **and**
    ($local\_open\_read_{kj} < global\_open\_write_j$)
      Discard the replica of O_j and inform $P(O_j)$;
  }
}

Figure 6. The EXIT Algorithm.

TIME CHECK Algorithm

When (a local time-check event occurs)
{
  For each local U_i
  {
    If (the on/off schedule status has changed)
    {
      call EXIT Algorithm for U_i;
      call ENTER Algorithm for U_i;
    }
  }
}

Figure 7. The TIME CHECK Algorithm.

EMERGENCY ENTER Algorithm

When (an emergency event occurs at B_k)
{
  For all O_j required in the emergency event
  {
    $local\_emergency\_counter_{kj}$ += 1;

    If (no local replica of O_j exist)
      Allocate a replica of O_j and inform $P(O_j)$;
  }
}

Figure 8. The EMERGENCY ENTER Algorithm.

EMERGENCY EXIT Algorithm

When (an emergency event is over at B_k)
{
  For all O_j required in the emergency event
  {
    $local\_emergency\_counter_{kj}$ -= 1;

    If ($local\_emergency\_counter_{kj} = 0$) **and**
    ($local\_open\_read_{kj} < global\_open\_write_j$)
      Discard the replica of O_j and inform $P(O_j)$;
  }
}

Figure 9. The EMERGENCY EXIT Algorithm.

(5) active replication algorithm. For each scheme, we measure the average access cost (both read and write), average response time (for read request), and average local availability of data. Both the access cost and response time are computed based on network distance. The local availability is the percentage of read access that can be satisfied locally. Because of the space limit, we only present the results on access cost and local availability.

We classify sites and data objects into classes. An *access class* is a set of objects that are routinely accessed by some fixed set of users. A *location class* is a set of sites that are constantly visited by some fixed set of users. Each user is assigned an access class and a location class to represent his/her access practice and mobility range. The variation of user behavior is controlled by the following parameters: (1) *movement locality* (the probability of moving within location class); (2) *access locality* (the probability of accessing within access class); (3) *schedule conformability* (the degree to which the user's actual behavior follows his/her schedule); (4) *write ratio* (the ratio of a user's write requests with respect to all requests). We have conducted a variety of simulation on different setting of the parameters. For each set of experiment, we varied one of the parameters while keeping others fixed so as to isolate the effect of the former.

5.1 Movement Locality

From Figure 10, it can be seen that the higher the locality the lower the average access cost. This is because when the users tend to move within a fixed area, most replicas are allocated around that area which results in lower access and replica maintenance cost. Higher movement locality also results in higher local availability (Figure 11). When the locality is low, however, our active replication scheme is the only one that can still maintains a good level of data availability. This is due to the use of schedule and predictive replication.

5.2 Access Locality

It can be observed from Figure 12 that the higher the access locality the lower the average access cost. Stronger locality implies a more stable access pattern. This is advantageous to all replication algorithms, even for the case with no replication (since the initial distribution of data is also made according to the access class partition). From Figure 13 we can see that except for no replication, all algorithms achieve higher availability when the users reveal stronger access locality. For both set of experiments, our algorithm performs consistently better than all other algorithms.

5.3 Schedule Conformability

From Figure 14 we can see that schedule conformability does not have significant impact on all five algorithms. This is conceivable for algorithms which do not employ schedule. For our algorithm, a user conforms to his/her schedule simply means he/she shows up at the right place and time. It does not place any constraint on the objects accessed by the user. Remote access is still allowed. Furthermore, access cost includes write cost. Therefore schedule conformability does not have a strong impact on access cost. The situation is different, however, for local availability (Figure 15). For other algorithms, schedule conformability has no effect. For active replication, we observed higher local availability with higher conformability. This is a successful demonstration of the benefit of using schedule.

5.4 The Effect of Write Requests

From Figure 16, all algorithms incur higher access cost when the write ratio is higher. The impact is especially evident on static replication. When the write ratio approaches 100%, the average access cost of static replication becomes very high since all replicas incur only consistency maintenance overhead. Dynamic replication schemes, however, are capable of discarding replicas that are no longer beneficial. When the write ratio approaches 0% (i.e. close to read-only access pattern), the cost calculation of three dynamic replication schemes almost always results in positive decision and thus a nearly full replication situation. In such case, most read requests can be satisfied locally, resulting in very low access cost. Local availability is also affected significantly (Figure 17). The higher the ratio

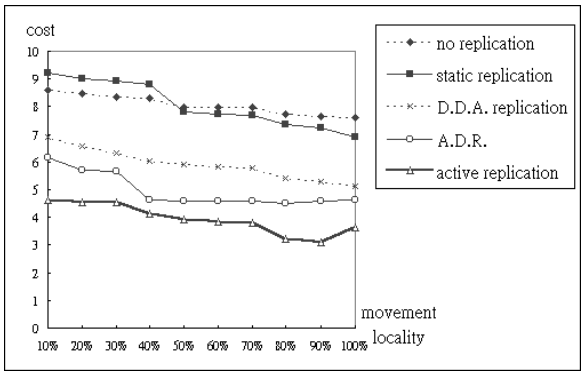

Figure 10. Average access cost of five replication schemes with varying movement locality.

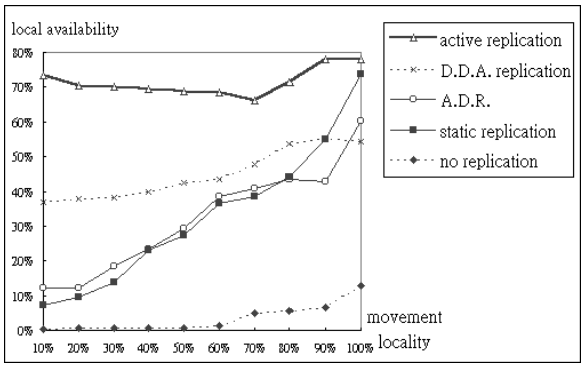

Figure 11. Average local availability of five replication schemes with varying movement locality.

the lower the availability since the reduction of replicas increases the percentage of remote access.

6. Conclusions and Future Work

Traditional replication schemes are passive in nature, and rarely consider the characteristics of mobile environment. We have proposed a dynamic replication scheme that actively provides replication services for mobile users. With more precise and responsive cost model, as well as the help of schedule, our scheme successfully demonstrates its ability to reduce access cost, improve response time, and achieve high local availability. The implementation of our scheme calls for careful consideration of several issues such as the maintenance of access histories and user profiles, the selection of primary copy, the limits on the number of replicas, clock synchronization for periodic time check events, and the possible adoption of other cost models. The perfor-

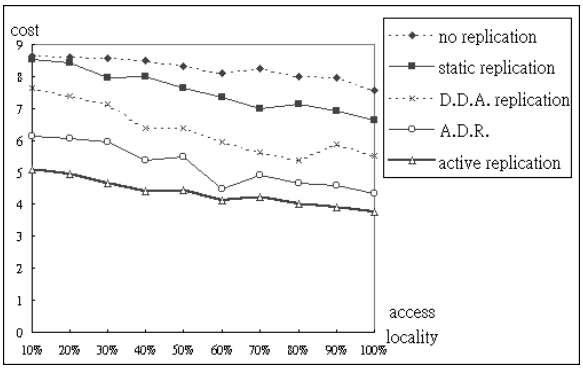

Figure 12. Average access cost of five replication schemes with varying access locality.

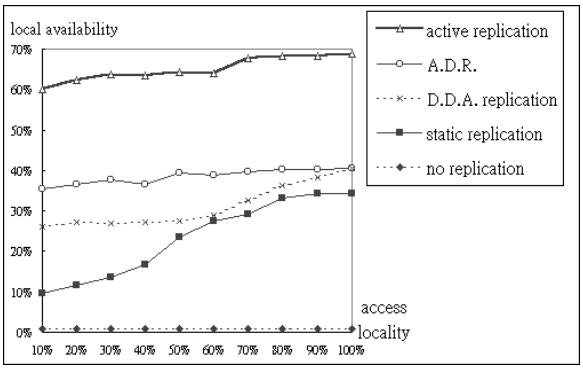

Figure 13. Average local availability of five replication schemes with varying access locality.

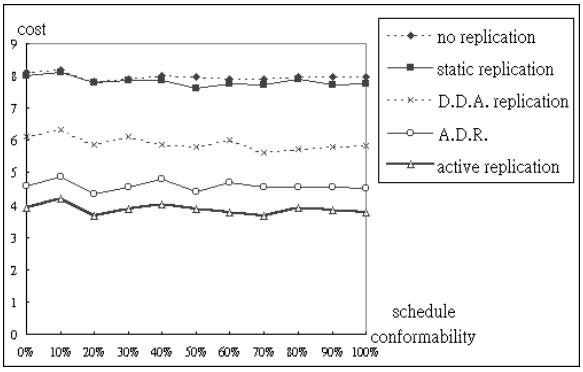

Figure 14. Average access cost of five replication schemes with varying schedule conformability.

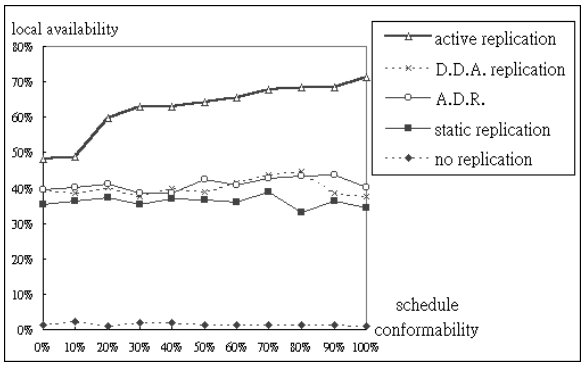

Figure 15. Average local availability of five replication schemes with varying schedule conformability.

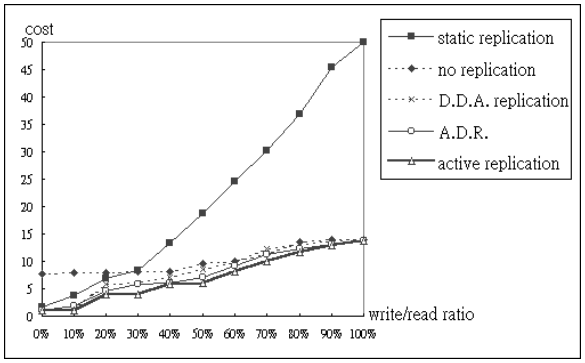

Figure 16. Average access cost of five replication schemes with varying write ratio.

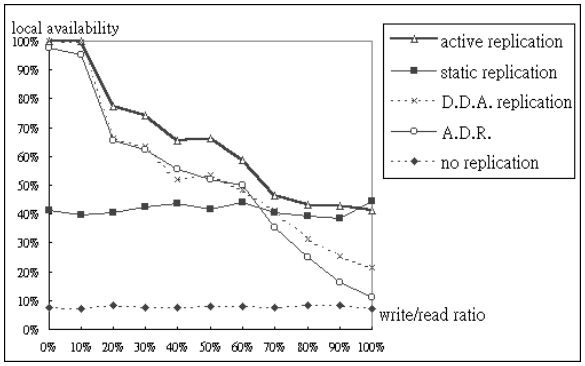

Figure 17. Average local availability of five replication schemes with varying write ratio.

mance of our algorithm can be further improved if the replication scheme is integrated with other mechanisms such as caching, prefetching, and data broadcasting.

References

[1] S. Acharya and S. B. Zdonik. An efficient scheme for dynamic data replication. Technical Report CS-93-43, Brown Univ CS Department, 1993.

[2] B. R. Badrinath and T. Imielinski. Replication and mobility. In *Proc. 2nd Workshop on the Management of Replicated Data*, pages 9–12, Monterey, CA., 1992.

[3] B. Ciciani, D. M. Dias, and P. S. Yu. Analysis of replication in distributed database systems. *IEEE Trans. on Knowledge and Data Engineering*, 2(2):247–261, 1990.

[4] R. David, R. Peter, and P. Gerald. Replication requirements in mobile environments. Technical Report 970021, UCLA, 1997.

[5] A. A. Helal, A. A. Heddaya, and B. B. Bhargava. *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers, 1996. ISBN: 0-7923-9800-9.

[6] Y. Huang, A. P. Sistla, and O. Wolfson. Data replication for mobile computers. In *SIGMOD Conference*, pages 13–24, 1994.

[7] S.-Y. Hwang, K. K. S. Lee, and Y. H. Chin. Data replication in a distributed system: A performance study. In *DEXA*, pages 708–717, 1996.

[8] M. C. Little and D. L. McCue. Computing replication placement in distributed system. In *Proc. IEEE 2nd Workshop on Replicated Data*, pages 58–61, Monterey, CA., 1992.

[9] M. C. Little and S. K. Shrivastava. Using application specific knowledge for configuring object replicas. In *Proc. IEEE 3rd Intl. Conf. on Configurable Distributed Systems*, May 1996.

[10] E. Pitoura. A replication schema to support weak connectivity in mobile information systems. In *DEXA*, pages 510–520, 1996.

[11] N. Shivakumar, J. Jannink, and J. Widom. Per-user profile replication in mobile environment: Algorithms, analysis, and simulation results. *MONET*, 2(2):129–140, 1997.

[12] J. Sidell, P. M. Aoki, A. Sah, C. Staelin, M. Stonebraker, and A. Yu. Data replication in Mariposa. Research Report S2K-95-62, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1995.

[13] O. Wolfson and S. Jajodia. Distributed algorithms for dynamic replication of data. In *PODS*, pages 149–163, 1992.

[14] O. Wolfson and S. Jajodia. An algorithm for dynamic data allocation in distributed systems. *Information Processing Letters*, 53(2):113–119, 1995.

[15] O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *ACM Trans on Database Systems*, 22(2):255–314, June 1997.