Transaction Based Dynamic Partial Replication in Mobile Environments

Manghui Tu, Peng Li, I-Ling Yen
Department of Computer Science
University of Texas at Dallas
{tumh2000, pengli, ilyen}@utdallas.edu

Abstract

In this paper, we investigate a transaction based partial replication strategy for replicating database in the mobile environments. We propose a heuristic algorithm to select a set of data objects for replicating to and de-allocating from the mobile node, based on the dynamically changing client access patterns. We conduct experiments to compare the performance of our algorithm with that of the full replication and frequency based partial replication algorithms. The results show that the transaction based partial replication algorithm always has a better performance than the full replication strategy. And, in most cases, it is better than frequency-based partial replication algorithms.

1 Introduction

Mobile computing has received a lot of attention in recent years. Technologies such as BlueTooth [4], iMobile [Che02], and Wi-Fi [8] have been rapidly deployed. Many distributed applications are being enhanced to support mobile clients. In most of these applications, it is common for the mobile nodes to access backend database and/or shared data files. However, due to the high communication cost and frequent disconnections in mobile environment, replication of data objects at the mobile nodes is needed to ensure the system availability, reliability, and performance [2] [13] [7].

Many research efforts have been devoted to the replication issues in mobile environment. Most of them focused on replica update protocols. Eager replication [3], which requires read one and write all in one transaction, is not suitable for mobile applications where most nodes are disconnected [6]. Some lazy update protocols were proposed for mobile systems. The two-tier lazy master replication scheme allows mobile nodes to read and update the database while disconnected from the network [6]. This scheme considers full replication at both mobile and stationary base nodes, and the master copy may be on the mobile node or on the base node. However, the actual decisions on where the master copies of the data objects should be have not been addressed. Epidemic update protocols are naturally suitable for mobile system [1]. In the epidemic update protocols, updates are executed locally and propagated to other nodes asynchronously by exchanging control information between neighboring nodes, and eventually all the replicas will converge to a consistent state. [7] proposed a family of epidemic algorithms for partially replicated systems.

Research works show that in distributed applications, client access patterns have some degree of locality when they repeatedly accessing the same set of services or data objects [10]. Replicating highly accessed services and data objects and place them close to the client can yield reduced communication cost and decreased network traffic. In this case, placement of replicated service and data objects are critical to system performance. Consider a mobile system with a base stationary node and some mobile nodes. If there are frequent queries from the mobile nodes and infrequent updates from base node, then replication of frequently queried data objects at the mobile nodes can reduce the communication overhead.

Traditionally, replica placement is done in a static fashion at the configuration time. The replication scheme remains unchanged until it is configured [5]. In a system with changing access patterns, static replica placement may severely degrade the benefit of replication. Recently, several works addressed the dynamic replica placement issues [14] [9]. The basic approach is to compute the replica locations to minimize the total access communication cost. However, most of these algorithms consider full replication, or partial replication of independent data objects. Full replication is not feasible in many applications due to the growing database size [7]. In the mobile environment, partial replication would be more realistic. More importantly, full replication may incur unnecessary overhead, for example, the data objects that are read frequently should have a higher degree of replication, while data objects that are updated frequently should have a low degree of replication or no replication. There are some works studying partial replication issues. In [12][11], databases are partitioned into clusters at configuration time, either for capacity or performance purpose. These strategies are static and they do not consider replication. [7] discusses an epidemic algorithm for update in partially replicated database, but does not consider partial replica placement. In [10], partial replication is considered for an information retrieval system. The documents are replicated at strategic sites based on the access frequency and site capacity. In [14], partial replication of independent data objects has been
discussed. But it cannot be used for applications such as transaction processing systems where dependency between data objects has significant impact to system performance.

In this paper, we address the placement issues for partial replication of dependent data objects. Data objects are dependent if they are accessed by the same transactions. If two data objects are accessed by the same read transactions, replicating both of them can reduce the communication overhead. If only one of them is replicated at the transaction-issuing site, there will be no benefit. For two data objects accessed by the same update transactions, replication of both will introduce same communication cost as the case where only one is replicated. We focus on mobile systems that consist of a stationary base node and a set of mobile nodes. Our goal is to improve the mobile nodes data accessing performance, especially to reduce the communication cost due to data accesses. We propose a data placement algorithm to dynamically replicate data objects, based on dynamically changing access patterns. In this paper, algorithms are given to find suitable set of data objects for replication and de-allocation.

The remainder of this paper is organized as follows. Section 2 describes our system model and problem definition at mobile nodes. Section 3 analyzes the cost and benefit models for replica allocation and de-allocation. Section 4 introduces the partial replication algorithm in details. The properties of our algorithm are discussed in section 5. Section 6 presents the experimental study and results of our algorithm. Section 7 states the conclusion of the paper.

2 System Model and Problem Specification

Consider a mobile system consisting of a stationary server node $P_{BC}$ and a set of mobile nodes. Node $P_{BC}$ hosts a database and mobile clients issue transactions to access the database. The database may be partially replicated at mobile nodes. Let $D_{BC} = \{d_1, d_2, \ldots, d_N\}$ denote the set of $N$ data objects that is available on $P_{BC}$. Let $2^{D_{BC}}$ denote the power set over $D_{BC}$. Assume that all mobile nodes are independent with each other, i.e. they directly interact with $P_{BC}$. Thus, without lose of generality, we can consider only one link between a mobile node $P_{MC}$ and $P_{BC}$. Let $D_{MC}$ denote the set of data objects on a mobile node $P_{MC}$. $D_{MC} \subseteq D_{BC}$. The power set over $D_{MC}$ is denoted as $2^{D_{MC}}$. Let $q(s)$ denote a read transaction issued by $P_{MC}$ and $w(s)$ an update transaction issued by any node other than $P_{MC}$, accessing all the data objects in $s$, where $s \subseteq 2^{D_{BC}}$.

Our goal is to minimize the traffic on the link between $P_{MC}$ and $P_{BC}$. For a read transaction $q(s)$, if $s \subseteq D_{MC}$, then $q(s)$ is served at $P_{MC}$ locally, otherwise $q(s)$ is forwarded to $P_{BC}$. For an update transaction issued by $P_{MC}$, no matter whether $s \subseteq D_{MC}$, it needs to be propagated to $P_{BC}$. For an update transaction $w(s)$ issued by any node other than $P_{MC}$, if $s \cap D_{MC} \neq \emptyset$, then $w(s)$ need to be propagated to $P_{MC}$. The traffic between $P_{MC}$ and $P_{BC}$ is measured by the number of messages between $P_{MC}$ and $P_{BC}$. We assume that for any transaction $q(s)$ or $w(s)$, if communication between $P_{BC}$ and $P_{MC}$ is required, then only one message is needed irrespective of the size of $s$.

To minimize the traffic on the link, we need to determine the set of data objects to be placed on $P_{MC}$. Since client access patterns show some locality, we can determine the best $D_{MC}$ based on the historical access patterns. Consider a time period $T$. $P_{BC}$ maintains a transaction log $L_{BC}$ and $P_{MC}$ maintains a transaction log $L_{MC}$. When $P_{BC}$ or $P_{MC}$ receives a transaction $q(s)$ or $w(s)$, a transaction record is made and appended to $L_{BC}$ or $L_{MC}$ respectively. Let $T_i$ denote a transaction record in the logs. $T_i$ includes two fields, $T_i$.type and $T_i$.dataObjectSet. $T_i$.type is the transaction type, e.g. read or update. $T_i$.dataObjectSet is the set of data object accessed, e.g., $s$ in $q(s)$ or $w(s)$. Let $Q(S_i)$ be the set of read transactions issued by $P_{MC}$ in the time duration $T_i$, accessing all the data objects in $S_i$, i.e. $Q(S_i) = \{q(s) | s = S_i\}$, $S_i \subseteq 2^{D_{MC}}$. Also, let $W(S_i)$ be the set of update transactions issued by any node other than $P_{MC}$ in $T_i$, accessing all the data objects in $S_i$, i.e. $W(S_i) = \{w(s) | s = S_i\}$, $S_i \subseteq 2^{D_{MC}}$. At the end of $T$, $P_{BC}$ executes log analysis algorithm. It computes $|Q(S_i)|$ and $|W(S_i)|$ for each $S_i$ accessed by transactions in $L_{BC}$. Then, the replica allocation algorithm determines the set of data objects to be replicated to $P_{MC}$ (whose read cost is less than update benefit). The set of data objects for replication is then sent to $P_{BC}$. Finally the log file is deleted. After $P_{BC}$ executing the allocation algorithm, $P_{MC}$ executes the replica de-allocation algorithm and de-allocates a set of data objects in a similar way (if $D_{MC} \neq \emptyset$).

The communication cost introduced by replica placement and de-allocation is not considered in this paper. Message losses and node failures are also beyond the scope of this paper.

3 Cost Analysis

Consider a set of data objects $R$, $R \subseteq 2^{D_{BC}}$. If $R$ is replicated at $P_{MC}$, a transaction $q(s)$ issued by $P_{MC}$, $s \subseteq R$, can be processed locally and one message can be saved. We consider this as one unit of benefit for replicating $R$ on $P_{MC}$. However, due to the replication of $R$ on $P_{MC}$, an update transaction $w(s)$ issued by a node other than $P_{MC}$, $s \cap R \neq \emptyset$, need to be propagated to $P_{MC}$. We consider this as one unit of cost introduced by replicating $R$ on $P_{MC}$.

We define update($R$) as the additional cost if $R$ is replicated to $P_{MC}$. Essentially, update($R$) is the number of update transactions accessing some data objects in $R$, issued by node other than $P_{MC}$. It can be expressed as:
\[ (1.1) \quad \text{update}(R) = \sum_j |W(j)|, \quad \text{where } (S_j \cap R \neq \phi) \land (S_j \cap D_{MC} = \phi) \]

We define \( \text{read}(R) \) as the additional benefit if \( R \) is replicated to \( P_{MC} \). Essentially, the \( \text{read}(R) \) is the number of read transactions accessing some data objects in \( R \), issued by node \( P_{MC} \). It can be expressed as:

\[ (1.2) \quad \text{read}(R) = \sum_j |Q(j)|, \quad \text{where } (S_j \subseteq (R \cup D_{MC})) \land ((S_j \cap R) \neq \phi) \]

Now, we define \( \text{cost}(R, D_{MC}) \) as the data object access cost for replicating \( R \) at \( P_{MC} \). It is the difference between update cost and read benefit for replicating \( R \) at \( P_{MC} \). It can be expressed as:

\[ (1.3) \quad \text{cost}(R, D_{MC}) = \text{update}(R) - \text{read}(R) \]

A set of data objects, \( R \subseteq D_{MC} \), may need to be de-allocated from \( P_{MC} \) if replicating \( R \) no longer brings benefit in terms of communication cost. Let \( \text{update}_d(R) \) denote the de-allocation benefit for de-allocating \( R \) and it is calculated as the same way we calculate \( \text{read}(R) \) in replica allocation analysis. Let \( \text{read}_d(R) \) denote the de-allocation cost and it is calculated the same way we calculate \( \text{update}(R) \) in replica allocation analysis. The de-allocation benefit of \( R \) includes all update transactions from \( P_{BC} \), accessing some data objects in \( R \).

\[ \text{update}_d(R) = \sum_j |W(j)|, \quad \text{where } (S_j \subseteq R) \land (S_j \subseteq D_{MC}) \land (S_j \neq \phi) \]

The de-allocation cost of \( R \) includes all read transactions issued by \( P_{MC} \), accessing some data objects in \( R \).

\[ \text{read}_d(R) = \sum_j |Q(j)|, \quad \text{where } (S_j \cap R \neq \phi) \land (S_j \subseteq D_{MC}) \land (R \subseteq D_{MC}) \]

The replica de-allocation access cost \( \text{cost}_d(R) \), is the difference between read cost and update benefit for de-allocating \( R \) from \( P_{MC} \).

\[ \text{cost}_d(R) = \text{read}_d(R) - \text{update}_d(R) \]

### 4 Replica Allocation and De-allocation Algorithms

The replica allocation and de-allocation algorithms include four components, including \( \log L_{BC} \) analysis, replica allocation, \( \log L_{MC} \) analysis, and replica de-allocation. To avoid transferring a large amount of log information, we separate the computation of replication allocation and de-allocation, and execute them on \( P_{BC} \) and \( P_{MC} \), respectively. We discuss these algorithms in the following subsections.

#### 4.1 \( \log L_{BC} \) Analysis

Let \( \overline{D_{MC}} = D_{BC} - D_{MC} \). Let \( 2^{|D_{MC}|} \) denote the power set over \( D_{MC} \). The summary transaction information from \( L_{BC} \) is recorded in \( S_{L_{BC}} \). \( S_{L_{BC}} \) consists of \( L \) entries, \( L \leq M \), one for each distinct data object set \( S_j \), \( S_j \in 2^{|D_{MC}|} \). Each entry contains three fields, the data set \( S_j \), \( |Q(S_j)| \), and \( |W(S_j)| \). Note that \( S_j \) is not necessarily the original data object set accessed by the transactions (will be discussed in the following).

Each transaction record \( T_r \) is extracted from \( L_{BC} \) and analyzed. Consider a read transaction \( q(s) \). The data objects not in \( s \cap D_{MC} \) will not affect the read benefit computation. So, only the data objects in \( s \cap D_{MC} \) need to be recorded in \( S_{L_{BC}} \). Thus, \( |Q(S)| \) is actually the number of read transactions \( q(s) \) where \( s \cap D_{MC} = S_j \). A recorded update transaction \( w(s) \) is relevant only if \( s \neq \phi \) and \( s \subseteq D_{MC} \). \( |W(S)| \) is then the number of update transactions accessing all data objects in \( S_j \). The pseudo code of the \( \log L_{BC} \) Analysis algorithm is shown in Fig. 1.

![Fig. 1. Pseudocode of the log \( L_{BC} \) analysis at \( P_{BC} \)](image)
4.2 Replica Allocation Algorithm

The replica allocation algorithm determines the best set of data objects to be replicated at \( P_{MC} \). Let \( \text{MinCost}(D_{MC}) \) denote the minimum cost for replicating a data object set to \( P_{MC} \). Then

\[
\text{MinCost}(D_{MC}) = \min_{j=1,2,...,\left|D_{MC}\right|} (\text{cost}(S_j, D_{MC})),
\]

where \( S_j \in 2^{D_{MC}} \).

If \( \text{MinCost}(D_{MC}) = \text{cost}(S_p, D_{MC}) < 0 \), then \( S_p \) should be allocated to \( P_{MC} \). Otherwise, if \( \text{MinCost}(D_{MC}) >= 0 \), no data object should be replicated.

However, this approach is infeasible. Suppose \( |D_{MC}| = K \leq N \). Although the size of \( S_i \) is \( L \), the size of all valid combinations of any non-empty subset of \( D_{MC} \) can be as big as \( 2^{N-K} \). This means that there may be \( 2^{N-K} \) entries involved in the \( \text{MinCost}(D_{MC}) \) computation. For every entry, we need to scan \( S_p \), which has a size of \( L \), the total computation complexity is \( L * 2^{N-K} \).

We develop a Bottom-up heuristic algorithm to determine a subset of the optimal replication data object set, as shown in Fig. 2. The Bottom-up algorithm essentially tries to find a data object set \( S_{opt} \) that any non-empty subset of \( S_{opt} \) will bring benefit if replicated at \( P_{MC} \). At each step, getNextMinimizedSet\( (S_{opt}) \) extract a data object set \( S_j \) from \( S_{BC} \), where \( |S_j| > |S_{opt}| \) is minimum among all remaining data object sets in \( S_{BC} \). If there are more than one data object sets with the same minimum size, then one of them is chosen arbitrarily. Thus only a minimum set of data objects is put into \( S_{opt} \) at each step, if it can bring benefit when replicated at \( P_{MC} \). We will show in section 5 that the Bottom-up algorithm obtains a subset of the optimal set. The worst case complexity of the algorithm is \( O(L^2N) \), where \( L \) is the size of \( S_i \). The Bottom-up heuristic algorithm assumes that the data object set accessed by the next transaction can determine the read benefit with the current \( S_{opt} \). However the assumption is not always true. For example, consider 6 read transactions on \( \{X, Y\} \), 7 read transactions on \( \{Y, Z\} \), and 10 update transactions on \( \{Y\} \). The Bottom-up algorithm will fail to replicate \( \{X, Y, Z\} \). Instead, it will claim that no object can be replicated at \( P_{MC} \). To solve this problem, we also consider an Up-down heuristic algorithm, which essentially tries not to replicate data objects that will not bring benefit if replicated at \( P_{MC} \).

The Up-down algorithm is similar to the Bottom-up algorithm and the pseudocode is shown in Fig. 3. \( S_{opt} \) obtained in Up-down algorithm is a superset of the optimal solution of the algorithm. The Up-down algorithm is not perfect either. For example, consider 6 update transactions on \( \{X, Y\} \), 7 update transactions on \( \{Y, Z\} \), 10 read transactions on \( \{Y\} \), and 5 read transactions on \( \{P\} \). The Up-down algorithm will try to replicate \( \{X, Y, Z, P\} \) onto the node \( P_{MC} \) although \( \{X, Y, Z\} \) will not bring any benefit. If we combine the two algorithms together and always select the better solution set, the results can be improved.

---

**Fig. 2. Bottom-up heuristic replica allocation algorithm**

1. \( S_{opt} \): replication data object set, initialized to \( \phi \).
2. \( S_j \): the next set in \( S_{BC} \) that makes the size of \( S_{opt} \cup S_j \) is minimum, initialized to \( \phi \).
3. Copy \( S_{BC} \) to \( S_{ij} \).
4. While \( S_j \neq \phi \):
   - \( S_j = \text{getNextMinimizedSet}(S_{opt}) \);
   - if \( \text{cost}(S_p, D_{MC}) < 0 \) \&\& \( S_j \cap S_{opt} = \phi \) then
     - \( S_{opt} = S_{opt} \cup S_j \);  
     - else if \( (\text{cost}(S_p, D_{MC}) >= 0 \&\& S_j \cap S_{opt} = \phi) || (S_j \subseteq S_{opt}) \) then
       - \( S_{opt} \) remains unchanged;
   - else if \( \text{cost}(S_{opt} \cup S_j, D_{MC}) < \text{cost}(S_{opt}, D_{MC}) \) then
     - \( S_{opt} = S_{opt} \cup S_j \);
   - delete \( S_j \) from \( S_{ij} \).
5. if \( \text{cost}(S_{opt}, D_{MC}) < 0 \) then \( S_{opt} \) is the set of data objects for replication to \( P_{MC} \).
The set data objects accessed by at least one transaction;

Copy $S_{BC}$ to $S_{t1}$;

While $S_{t1} \neq \emptyset$

$S_j = \text{getNextMinimizedSet}(S_{opt})$;

if $\text{cost}(S_j, D_{MC}) \leq 0$ & $S_j \cap S_{opt} = \emptyset$ then

$S_{opt} = S_{opt} \cup S_j$;

else if $\text{cost}(S_j, D_{MC}) > 0$ & $S_j \cap S_{opt} = \emptyset$ & ($S_j \subseteq S_{opt}$) then

$S_{opt}$ remains unchanged;

Else if $\text{cost}(S_{opt} \cup S_j, D_{MC}) \leq \text{cost}(S_{opt}, D_{MC})$ then

$S_{opt} = S_{opt} \cup S_j$;

delete $S_j$ from $S_{t1}$.

$S_{opt} = S_{used} - S_{opt}$;

if $\text{cost}(S_{opt}, D_{MC}) < 0$ then

$S_{opt}$ is the set of data objects for replication to $P_{MC}$.

**Fig. 3.** Up-down heuristic replica allocation algorithm

### 4.3 Log $L_{MC}$ Analysis

$S_{t_{MC}}$, a data structure similar to $S_{BC}$ is temporarily maintained on $P_{MC}$. $S_{t_{MC}}$ records the summary transaction information from Log $L_{MC}$. Consider an update transaction $w(s)$. The data objects in $s \cap D_{MC}$ will not affect the update benefit computation. So only the data objects in $s \cap D_{MC}$ need to be recorded in $S_{t_{MC}}$. Thus, $|W(S_j)|$ is actually the number of update transactions $w(s)$ where $s \cap D_{MC} = S_j$. A recorded read transaction $q(s)$ is said to be relevant only if $s \neq \emptyset$ and $s \subseteq D_{MC}$. The pseudo code of the Log $L_{MC}$ Analysis is shown in Fig. 4.

### 4.4 Replica De-allocation Algorithm

The optimal set of data objects for replica de-allocation is determined by finding a subset of $D_{MC}$ that produces the $\text{MinCost}()$. Then

$$\text{MinCost}() = \min_{d \in 2^{D_{MC}}} (\text{cost}_d(S_j)), \text{ where } S_j \in 2^{D_{MC}}$$

If $\text{MinCost}(d(S_j)) < 0$, then $S_j$ on $P_{MC}$ should be de-allocated. If no such a subset of $D_{MC}$ exists, then no data object can be de-allocated from $P_{MC}$.

The same problem as that in replica allocation occurs here. The straightforward approach is not feasible for large $N$ due to the high computation complexity. Since the two problems are similar, we can use the replica allocation heuristic algorithms to get a subset of the optimal set of data objects for de-allocation. The difference is only the benefit and cost functions. We will not go into the details of the algorithm here.

**Fig. 4.** Pseudo code of log $L_{MC}$ analysis on $P_{MC}$.
5 The Properties of the Algorithm

Here we show that the Bottom-up heuristic algorithm gives a subset of the optimal set for replica placement. The data object set obtained from the combined algorithm always introduces less or equal communication cost than the full replication. Note that $S_{opt}$ is the data object set obtained by replica allocation algorithm. Let $S'_{opt}$ denote the replication data object set obtained by previous step. Let $S_{opt_i}$ denote the optimal data object set. Let $R$ denote the next replication candidate data object set extracted from $S_i$. We first show in lemma 1, 3, and 4 that $S_{opt}$ obtained at each step will reduce communication cost, if $S_{opt}$ is replicated at $P_{MC}$. Lemma 2 shows that each data object in $S_{opt}$ makes positive contribution on the reduction of communication cost. Lemma 5 shows that both $S_{opt}$ and $S_{opt_{-i}}$ must be the union of the data object sets accessed by the read transactions counted as replication cost.

Lemma 1. If cost $(R, D_{MC}) < 0$ and $R \cap S_{opt} = \phi$, then $S_{opt} = S'_{opt} \cup R$.

Proof: We first show that read $(R) + read (S'_{opt}) \leq read (S'_{opt} \cup R)$.

According to Equation 1.1, read $(R) + read (S'_{opt})$

$$= \sum \langle Q(S_j) \rangle \text{where}\ (S_j \subseteq (R \cup D_{MC})) \land (\langle S \cap S'_{opt} \rangle \neq \phi)$$

$$\land (\langle S \cap (S'_{opt} \cup R) \rangle \neq \phi)$$

$$= \sum \langle Q(S_j) \rangle \text{where}\ (S_j \subseteq (S'_{opt} \cup R) \neq \phi) \land (\langle S \cap S'_{opt} \rangle \neq \phi)$$

$$\leq \sum \langle W(S_j) \rangle \text{where}\ ((S_j \cap (R \cup S'_{opt}) \neq \phi) \land (S_j \cap D_{MC} = \phi)$$

$$= \sum \langle W(S_j) \rangle \text{where}\ ((S_j \cap R \neq \phi) \lor (S_j \cap S'_{opt} \neq \phi)) \land ((S_j \cap R \neq \phi)$$

$$\land (S_j \cap D_{MC} = \phi)$$

$$= update (S'_{opt}) + update (R).$$

According to Equation 1.2, cost $(S'_{opt} \cup R, D_{MC}) = update (S'_{opt} \cup R) - \text{read} (S'_{opt} \cup R)$.

$$= \sum \langle W(S_j) \rangle \text{where}\ ((S_j \cap (R \cup S'_{opt}) \neq \phi) \land (S_j \cap D_{MC} = \phi)$$

$$= \sum \langle W(S_j) \rangle \text{where}\ ((S_j \cap R \neq \phi) \lor (S_j \cap S'_{opt} \neq \phi)) \land ((S_j \cap R \neq \phi)$$

$$\land (S_j \cap D_{MC} = \phi)$$

$$\leq update (S'_{opt}) + update (R).$$

Since cost $(R, D_{MC}) < 0$ and cost $(S'_{opt}, D_{MC}) < 0$, then we get the following two expressions to be true.

Lemma 2. Let $d$ be an arbitrary data object, $d \in S_{opt}$.

Then cost$(S_{opt}, D_{MC}) \geq cost (S_{opt} - \{d\}, D_{MC})$.

Proof: Let $R$ denote the data object set extracted from $S$, $d \in R$, $d \notin S'_{opt}$ and cost$(S_{opt} \cup R, D_{MC}) < cost (S'_{opt}, D_{MC})$.

Let define $S_i = R - \{d\}$, $R \notin S_{opt}$.

Let us $(R, S'_{opt})$

$$= \sum \langle W(S_j) \rangle \text{where}\ (S_j \cap D_{MC} = \phi) \land (S_j \cap S'_{opt} = \phi) \land (S_j \cap R \neq \phi)$$

and $r(R, S'_{opt})$

$$= \sum \langle Q(S_j) \rangle \text{where}\ (S_j \subseteq (R \cup S'_{opt} \cup D_{MC})) \land (S_j \cap R \neq \phi)$$

If $S_j = \phi$, then $R \in \{d\}$, $R \cap S'_{opt} = \phi$, cost $(\{d\}, D_{MC}) < 0$.

So $u(\{d\}, S_{opt}) \leq u(R) < \text{read}(R) = \{Q(\{d\})\}$.

Then $|Q(\{d\})| - |u(\{d\}, S_{opt})| < 0$, and cost $(S_{opt} - \{d\}, D_{MC}) - cost (S_{opt}, D_{MC}) \geq |Q(\{d\})| - u(\{d\}, S'_{opt})$.

If $S_j \neq \phi$ and $R \cap S'_{opt} = \phi$, then cost $(R, D_{MC}) < 0$, so $u(R, S'_{opt}) \leq u(R) < \text{read}(R)$.

According to allocation algorithm, $S_i \notin S'_{opt}$ means update$(S_i) \geq \text{read}(S_i)$, $u(\{d\}, S_i) + \text{update}(S_i) = \text{update}(R)$, and $r(\{d\}, S_i)$ + read$(S_i) = \text{read}(R)$, so $u(\{d\}, S_i) < r(\{d\}, S_i)$.

According to equation 1.1 and equation 1.2, we can easily get $u(\{d\}, S_{opt} - \{d\}) \leq u(\{d\}, S_i) + r(\{d\}, S_i) \leq r(\{d\}, S_{opt})$, and $u(\{d\}, S_{opt} - \{d\}) + \text{update}(S_{opt} - \{d\}) = \text{update}(S_{opt})$, $r(\{d\}, S_{opt}) + \text{read}(S_{opt} - \{d\}) = \text{read}(S_{opt})$.

According to equation 1.3, it can be concluded that cost$(S_{opt}, D_{MC}) < cost (S_i - \{d\}, D_{MC})$.

Let $S_i = S_k - (S_k \cap S'_{opt})$. If $S_i \neq \phi$, $S_k \cap S'_{opt} \neq \phi$. Since cost$(S'_{opt} \cup R, D_{MC}) < cost (S'_{opt}, D_{MC})$, cost$(S'_{opt} \cup R, D_{MC}) < cost (S_{opt}, D_{MC})$, $u(\{d\}, S_{opt} - \{d\}) - r(\{d\}, S_{opt}) \leq 0$.

Ang the allocation algorithm, $S_i \cap S'_{opt} = \phi$, we can get $u(\{d\}, S_{opt} - \{d\}) - r(\{d\}, S_{opt}) \geq 0$.

We then get cost$(S'_{opt} \cup S_0, D_{MC}) \geq cost (S'_{opt}, D_{MC})$.

Then $u(\{d\}, S_{opt} - \{d\}) \leq r(\{d\}, S_{opt})$.

So $u(\{d\}, S_{opt} - \{d\}) \leq r(\{d\}, S_{opt})$, thus cost$(S_{opt}, D_{MC}) < cost (S_{opt} - \{d\}, D_{MC})$.

From above, we can conclude if $d \in S_{opt}$ then cost$(S_{opt}, D_{MC}) < cost (S_{opt} - \{d\}, D_{MC})$ is true. €

Lemma 3. If $R \notin S'_{opt}$, then $S_{opt} = S'_{opt}$.

Proof: Here we need to show if a candidate replication set $R$ is a subset of $S'_{opt}$ then no change will be made to $S'_{opt}$. Obviously if $R = S'_{opt}$ then $S_{opt} = S'_{opt}$. Consider the case $R \subset S'_{opt}$.
Let \( R \cup S_k = S'_{opt}, R \cap S_k = \emptyset \). According to Lemma 2, any data object set \( d \in S_k \) cannot be taken out from \( S'_{opt} \), so \( S_k \) cannot be taken out from \( S'_{opt} \). Thus \( S_{opt} = S'_{opt} \in \mathcal{E} \).

**Lemma 4.** If \( \text{cost}(S'_{opt} \cup R, D_{MC}) < \text{cost}(S'_{opt}, D_{MC}) \), then \( S_{opt} = S'_{opt} \cup R \).

Proof: If \( S'_{opt} \cup R = R \), then \( S_{opt} = S'_{opt} \cup R \). Otherwise, according to Lemma 2, any data object cannot be taken out from \( S'_{opt} \). Let \( S_k = R - (S'_{opt} \cap R) \), then \( S_k \cap S_{opt} = \emptyset \). According to Lemma 1, \( S_{opt} = S'_{opt} \cup S_k = S'_{opt} \cup R \). \( \in \mathcal{E} \).

**Lemma 5.** \( S_{opt} \) obtained by the replication allocation algorithm and the optimal set \( S_{opt,e} \) must be the union of data object sets accessed by read transactions that are counted as benefit.

Proof: For any data object \( d \in S_{opt} \), \( d \) must have been accessed by at least one read transaction \( q(s) \). \( q(s) \) is counted as the read benefit. Otherwise, the cost on \( d \) will not be less than 0. According to the cost model, \( q(s) \) is counted as read benefit only when \( s \subseteq S_{opt} \). The same can be applied to \( S_{opt,e} \).

**Theorem 1.** The Bottom-up heuristic algorithm obtains a subset of the optimal set for replica allocation.

Proof: Assume \( S_{opt,e} \cap S_{opt} = \emptyset \). According to Lemmas 1, 2, 5, this is impossible because \( \text{cost}(S_{opt,e} \cup S_{opt}) < \text{cost}(S_{opt,e}) \).

Assume \( S_{opt,e} \cap S_{opt} = S_{opt} - S_{opt} \neq \emptyset \). Let \( S_1 = S_{opt} - S_{opt} - S_2 = S_{opt,e} \cap S_{opt} \). Thus \( S_1 \cap S_2 = \emptyset \). According to Lemmas 1, 2 and 5, this is impossible because \( \text{cost}(S_{opt,e} \cup S_{opt}) < \text{cost}(S_{opt,e}) \).

Assume \( S_{opt,e} \subseteq S_{opt} \), then according to Lemmas 1, 2 and 5, this is impossible because \( \text{cost}(S_{opt}) < \text{cost}(S_{opt,e}) \).

From above, we can conclude that \( S_{opt} \subseteq S_{opt,e} \). \( \in \mathcal{E} \).

**Theorem 2.** The combined heuristic algorithm is always better than or equivalent to the full replication.

Proof: The Up-down algorithm removes a subset of data objects of full database, and that set of data objects will introduce more cost than benefit for replication. So the communication cost introduced by the final set computed by the Up-down algorithm, is always smaller than or equal to the full replication. The final results for the combined algorithm select a data object set that introduces less or equal communication cost than the Up-down algorithm data object set. So the combined heuristic algorithm is always better than or equivalent to the full replication. \( \in \mathcal{E} \).

6 Experimental Study

We conduct experimental study to compare the performance of our transaction based partial replication allocation strategy with full replication strategy and frequency based partial replication strategy. For full replication strategy, the algorithm replicates the full set of data objects on node \( P_{BC} \) to node \( P_{BC} \), if the total number of read transactions is greater than that of the update transactions. For frequency based partial replication strategy, data objects accessed by more read transactions than update transactions are replicated to \( P_{MC} \). For our algorithm, we only consider the replica allocation algorithm, because the replica de-allocation algorithm works in the same way as the replica allocation algorithm.

For performance comparisons, we consider the number of messages saved by different replication algorithms. We use \( \text{Msg-Full}, \text{Msg-Freq} \) and \( \text{Msg-Tra} \) to represent the number of messages saved by the full replication, frequency based partial replication, and transaction based partial replication algorithms, respectively.

6.1 Transaction Generation

We consider the following parameters for transaction generation: 1) \( Z_S \), the skew of number of data objects accessed by transactions, 2) \( Z_D \), the skew of which data objects to be accessed, 3) \( T_N \), the maximum number of data objects accessed by one transaction, 4) \( N_S \), the total number of data objects on the node \( P_{BC} \), 5) \( T_N \), total number of transactions, 6) \( R/W \): the ratio of the read transactions issued by \( P_{MC} \) and update transactions issued by nodes other than \( P_{MC} \). To simulate the read and update transaction pattern in the mobile system, we use Zipf distribution to simulate the distribution of the number of data objects involved and which data objects to be accessed in each transaction. The skew of number of data objects involved in transactions is determined by the Zipf parameter \( Z_S \). Let \( t \) denote the number of data objects involved in a transaction, then the occurrence frequency of transactions whose size is \( s \) is inversely proportional to \( t \). The Zipf parameter \( Z_D \) determines the skew of which data objects to be accessed. We first order the data objects so that each data object is assigned a rank \( r \). Then the occurrence frequency of a data object in transactions is inversely proportional to its rank as \( 1/r^Z_D \). We conduct each experiment 1000 times and compute the average performance for comparison.

6.2 Total Number of Data Objects on Node \( P_{BC} \)

In this section, we measure the effect of the number of data objects on \( P_{BC} \) (\( N_S \)) to the performance of the three algorithms. The parameter settings are as follows: \( Z_D=1 \), \( Z_S=1 \), \( R/W = 1 \), \( T_S = 4 \), \( T_N = 1000 \). \( N_S \) is changed from 50 to 500. Fig. 5 shows message saving by the three replication strategies. With the increase of \( N_S \), full replication shows little variation on message saving. The performance of our algorithm improves when \( N_S \) increases. This is because larger data object sets are read and updated infrequently.

As can be seen from Fig. 5, our algorithm is always better than full replication when \( N_S > 100 \). Frequency based replication incurs increasing communication cost. This is because in many cases, only a part of data objects in read transactions is replicated and communication is still required to access the remaining part. When \( N_S \) increases, the negative effective reduces.
Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS’04)

becomes smaller with increasing lower communication cost. However, the reduction results are shown in Fig. 6. As \(|TS|\) number of data objects in a transaction is always equal to \(|D_{BC}|\), then our algorithm is the same as full replication. To measure the effect of the number of data objects in transactions, we set the parameters as follows: \(Z_D = 1, Z_S = 1, R/W = 1, N_S = 200, T_N = 1000\). The number of data objects in transactions, \(T_S\), is changed from 1 to 10. The results are shown in Fig. 6. As \(T_S\) increases, full replication shows little variation, but our algorithm and frequency based replication algorithm have degraded performance. Frequency based replication has degraded performance when \(T_S \geq 3\). Our algorithm always has a lower communication cost. However, the reduction becomes smaller with increasing \(T_S\).

Fig. 6. Effect of number of data objects in transactions.

6.3 Maximum Number of Data Objects in Transactions

The number of data objects in transactions is expected to be an important factor for determining the performance of the algorithms. If the number of data objects in a transaction is 1, then transaction based partial replication is the same as frequency based partial replication. If the number of data objects in a transaction is always equal to \(|D_{BC}|\), then our algorithm is the same as full replication. To measure the effect of the number of data objects in transactions, we set the parameters as follows: \(Z_D = 1, Z_S = 1, R/W = 1, N_S = 200, T_N = 1000\). The number of data objects in transactions, \(T_S\), is changed from 1 to 10. The results are shown in Fig. 6. As \(T_S\) increases, full replication shows little variation, but our algorithm and frequency based replication algorithm have degraded performance. Frequency based replication has degraded performance when \(T_S \geq 3\). Our algorithm always has a lower communication cost. However, the reduction becomes smaller with increasing \(T_S\).

Fig. 7. The effect of read/update ratio on message saving.

6.4 Read Update Ratio

In this section, we compare the performance of various algorithms with different read/update ratio. The parameter settings are as follows: \(Z_D = 1, Z_S = 1, N_S = 200, T_S = 2, T_N = 1000\). We change the read/update ratio \(R/W\) from 0.4 to 4. The performance results are shown in the

Fig. 7. When \(R/W\) increases, number of message saved by all three replication strategies increases. When the number of read transactions is higher than the number of update transactions, more data objects can be replicated and more communication messages can be saved. The improvement ratio decreases when \(R/W\) is greater than 1. With higher \(R/W\), more and more data objects can be replicated. So, there is a higher chance that the frequently referenced data objects will actually bring benefit to the system. The data objects replicated by the three replication strategies will converge to the same set of data objects.

6.5 Performance Impact by Other Parameters

Note that \(Z_D\) determines the access frequency of data objects in transactions, and \(Z_S\) determines the number of data objects in transactions. We first measure the performance impact due to different access frequencies. We set \(Z_S = 1, R/W = 1, N_S = 100, T_S = 2, T_N = 500\). We change \(Z_D\) from 0.2 to 2.0. The performance results are shown in Fig. 8. Compare to full replication and frequency based partial replication, our algorithm has better performance when \(0.6 \leq Z_D \leq 1.8\). When \(Z_D\) is small, the data objects accessed by transactions are more sparsely distributed. Thus, full replication and partial replication show similar behavior. When \(Z_D > 1.8\), number of data objects accessed by transactions are smaller. Since \(R/W\) is 1, read and update transactions on this set of data objects are more likely to be the same. Hence, the chance to save messages on this set of data objects is less if it is replicated. The effects of \(Z_D\) on the two partial replication strategies are similar. For the two partial replication strategies, the number of messages saved peaks when \(Z_D = 0.4\). As \(Z_D\) increases (for \(Z_D > 0.6\)), the performance of both partial replication strategies degrade. The numbers of message saved by all three replication strategies are low, because the read/update ratio is 1.

We now measure the performance impact of \(Z_S\). We set \(Z_D = 1, R/W = 1, N_S = 50, T_S = 3, T_N = 500\). \(Z_S\) is changed from 0.2 to 2.0. The performance results are shown in Fig. 9. With increasing \(Z_S\), the average number of data objects in transactions becomes smaller. More transactions tend to access a smaller number of data.
objects. So, the results show a similar effect as when a smaller number of data objects are accessed in transactions. As $Z_S$ increases, the two partial replication algorithms show better message savings.

![Fig. 8. Effect of the $Z_S$ on message saving.](image)

![Fig. 9. Effect of the $Z_S$ on message saving.](image)

7 Conclusion

In this paper, we propose a novel transaction based partial replication strategy for data object placement. An $O(M/MN)$ heuristic algorithms is given to partially replicated data objects to the mobile nodes, based on client access patterns derived from $M$ transactions recorded in a log. Our algorithm consists of Bottom-up and Up-down sub-algorithms. We prove that the Bottom-up algorithm always computes a subset of the optimal set, while the Up-down algorithm gives a superset of the optimal set. The combined algorithm of transaction based partial replication strategy, always show better or equal performance compare to the full replication algorithm.

We also conduct a series of experiments to evaluate the effects of different parameters. Results show that all parameters have significant effect on the replication performance. In most cases, transaction based partial replication strategy yields better performance than the full replication and frequency based partial replication algorithms.

References


