

Data Duplication and Consistency in a Mobile, Multidatabase Environment

J.B. Lim and A.R. Hurson
Dept. of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
hurson@cse.psu.edu

Abstract

As technological advances are made in software and hardware, the feasibility of accessing information “anytime, anywhere” is becoming a reality. There are similarities involved in accessing information in a multidatabase environment and a mobile computing environment. We have discussed the characteristics of a wireless-mobile computing environment superimposed on a multidatabase system to realize a system capable of effectively accessing a large amount of data over a wireless medium in [18]. In order to limit the effects of the restrictions imposed by a mobile environment, data duplication is used at the mobile unit. The duplicated data at the mobile node provide additional availability in case of a weak connection or disconnection. In such an environment, it is important to retain the consistency between the data residing on fixed nodes and mobile nodes. Current research has concentrated on page and file based caching or replication schemes to address the availability and consistency issues in a mobile environment. In a mobile, multidatabase environment, local autonomy restrictions prevent the use of a page or file based data duplication scheme. We propose a new data duplication scheme to address the limited bandwidth and local autonomy restrictions in such an environment. Queries and the associated data is cached at the mobile unit as a complete object. Consistency is maintained by using a parity based invalidation scheme. Additionally, a simple prefetching scheme is used in conjunction with caching to further improve effectiveness of the proposed scheme.

1. Introduction

A multidatabase management system (MDBMS) is intended to provide timely and reliable access to a large amount of global information. It is a global system layer that allows distributed access to multiple preexisting, autonomous, and heterogeneous local database management systems (DBMS). The global layer provides full database functionality and interacts with the local DBMSs at the external user interface. Both the hardware and software intricacies of the different local systems are transparent to the user, and access to different local systems appears to the user as a single, uniform system. Current multidatabase systems are designed to allow timely and reliable access to data in an environment that is characterized as “sometimes, somewhere.” Researchers for multidatabase systems have addressed issues such as autonomy, heterogeneity, transaction management, concurrency control, transparency, and query resolution. However, the diversity of the access devices and communication mediums introduces additional complexities and restrictions in a multidatabase system. Introduction of new technology has led to an extension of the traditional MDBMS model to include remote data access [18]. Remote access to data refers to both mobile nodes and fixed

nodes accessing a variety of data via a network connection that provides varying, intermittent, and weak connectivity. A mobile node introduces additional complexities such as location dependence, system configuration variations, and data placement issues. However, regardless of the hardware device, connection medium, and type of data accessed, all users share the same requirements: timely and reliable access to various types of data. This new environment is referred to as a mobile data access system (MDAS), and is composed of a wireless-mobile computing environment superimposed on a multidatabase system [18].

Naturally, many issues in an MDAS have their roots in a traditional multidatabase system and a conventional mobile environment. The following are two that are paramount to the consistent duplication of data:

- **Local Autonomy:** Each local DBMS maintains complete control over local data and resources. Modifications to the global system and global operations should not impact the operation of the local system.
- **Communication Restrictions:** The presence of a wireless communication link introduces issues such as lower bandwidth, weak connections, and disconnection.

In order to mitigate the effects of weak communications or disconnection, natural byproducts of mobility and wireless communication, some form of data duplication is used at the mobile unit. The duplicated data at the mobile node provide additional availability in case of a weak connection or disconnection. In such an environment, it is important to effectively address transactional consistency of the data between a fixed node and a mobile node. Researchers have extensively studied caching and replication schemes that may be used to address the constraints of a wireless link [3,7,11,15,20,24,27]. Current distributed database replication schemes are not suitable for an MDAS environment because consistency can not be efficiently maintained due to local autonomy requirements, communication constraints, and/or the required use of compensating transactions. Proposed caching schemes in distributed databases are also not suitable for an MDAS environment. Mainly due to the autonomy requirements of the local DBMS, the local information about the validity of a page or file in the local system is not available globally. Accordingly, any type of invalidation, polling, or timestamp-based method would be too impractical and inefficient to use, and in many cases, impossible.

In our proposed data duplication scheme, we address the limited bandwidth and local autonomy restrictions by using automated queued queries (AQ2) and caching of data in the form of a bundled query (BUNQ). To maintain data consistency for a traditional mobile user and mobile access device, read-only queries are cached as a bundled query and are validated using a

simple parity-checking scheme. We assume that only a small number of users will have write privilege to the MDAS, and that the majority of the accesses will be in the form of read-only requests. In addition, guaranteeing write consistency while disconnected is extremely difficult or not possible at all due to local autonomy requirements in an MDAS. Consequently, any transactions containing write operations are directly submitted to the MDAS system, or are queued during disconnection, and submitted upon re-connection.

The remainder of this paper is organized as follows: section 2 discusses the necessary background information and database model including the MDAS environment, distributed caching, and distributed prefetching. Section 3 addresses the proposed data duplication model and presents protocols for maintaining consistency in the MDAS environment. In section 4, we introduce a simulation model to test the effectiveness of the proposed scheme. Subsequently, the simulation results are presented and discussed. Section 5 concludes the paper and suggests some future directions for research.

2. Background

The following section briefly introduces the MDAS environment and discusses various issues involved in providing multidatabase functionality to a mobile user. Furthermore, the details of a multidatabase system, the summary schemas model (SSM), are discussed. The current SSM model is extended to incorporate data duplication for mobile units in an MDAS environment. The interested reader is referred to [6] and [18] for a more detailed discussion of the SSM and an MDAS environment.

2.1 Multidatabase Environment

A multidatabase system is a global system layer that allows distributed access to multiple preexisting local database systems. One of the essential features of a multidatabase system is local site autonomy, where each local database management system (DBMS) maintains complete control over local data and resources. The operation of the local system should not be affected by modifications to the global system and any global operations. There are three different forms of autonomy: Design Autonomy, Communication Autonomy, and Execution Autonomy.

In addition to autonomy, heterogeneous data access is an important aspect of a multidatabase system. Multidatabase systems are designed to access a multitude of data, irrespective of the system, data type, or particular local access method. There are two types of heterogeneity, software and hardware. Support for heterogeneity is a trade-off between developing and making changes in both hardware and software, and limiting participation. Consequently, as the number of systems and the degree of heterogeneity among these systems increases, the cost of integration into the global MDBMS increases. Furthermore, the global system software must also support any local deficiencies. Autonomy and heterogeneity leads to issues in schema integration, query languages, query processing, and transaction management [5].

2.2 Summary Schemas Model(SSM) for Multidatabase Systems

Accessing a heterogeneous multidatabase system is a

challenging problem. Multidatabase language and global schema systems suffer from inefficiencies and scalability problems. The SSM has been proposed as an efficient means to access data in a heterogeneous multidatabase environment [6]. The SSM uses a hierarchical meta structure that provides an incrementally concise view of the data in the form of summary schemas. The hierarchical data structure of the SSM consists of leaf nodes and summary schema nodes. Each leaf node represents a portion of a local database that is globally shared. The summary schema nodes provide a more concise view of the data by summarizing and integrating the schemas of its children nodes using synonym, hypernym, and hyponym links among schema terms.

The SSM intelligently resolves user queries [6]. When the user knows the precise access terms of a local database, the query is directly submitted to the corresponding local data source. However, when the precise access terms and/or location of the data is not known to the user (imprecise query), the SSM resolves the query in the user's own terms.

A simulation and benchmark of the SSM model was performed and the benefits are shown in [6,8]. The simulator compared and contrasted the costs of querying a multidatabase system using precise queries (access terms and locations are known) against the intelligent query processing capability of the SSM for the imprecise queries. The results showed that the intelligent query processing (imprecise query) of the SSM and an exact query (precise query) incurred very comparable costs (i.e. there were only small overhead costs involved). Interestingly, using intelligent SSM query processing actually outperformed an exact query in certain circumstances. These results are very relevant to an MDAS in that a user would most likely require this functionality while accessing heterogeneous data sources.

The SSM provides several benefits to the traditional multidatabase systems that can be directly applied to an MDAS. These include:

- The SSM provides global access to data without requiring precise knowledge of local access terms or local views. The system can intelligently process a user's query in his/her own terms.
- The SSM's hierarchical structure of hypernym/hyponym relationships produces incrementally concise views of the global data. The overall memory requirements for the SSM, compared to the requirements of a global schema, are drastically reduced by up to 94%. Subsequently, the SSM could be kept in main memory, thus reducing the access time and query processing time. Furthermore, for very resource limited devices in an MDAS, only portions of the upper levels of the SSM meta data structure could be stored locally, which would still provide a global view (albeit less detailed) of the data.
- The SSM can be used to browse/view the global data. The user can either 1) follow semantic links in a summary schemas node, or 2) query the system for terms that are similar to the user's access terms. In either case, the SSM could be used to browse data by "stepping" through the hierarchy, or view semantically similar data through queries.

2.3 The MDAS Environment

In addition to the issues involved in an MDBMS, accessing a large amount of data over a network connection of an MDAS involves the mobile networking environment and mobility issues.

The mobile environment includes the physical network architecture and access devices. Mobility issues involve the ability of the system to adapt to the additional restrictions imposed by a mobile environment.

In comparison to traditional fixed LANs, the network connections in a mobile environment have lower bandwidths, higher error rates, and are subject to frequent disconnection. Furthermore, due to large variations in bandwidths, there are varying degrees of disconnection that can occur with these types of networks. The type of hardware devices used in an MDAS is more diverse than a traditional system. These can include desktop computing devices, portable computers, and portable hand-held devices. The requirements and limitations of these devices vary greatly in terms of computing power, memory capacity, storage space, size, display area and energy consumption [18].

With the introduction of mobility, an application must be able to adapt to changing conditions in the environment [26]. These changes include the network environment and resources available to the application. A balance between self-autonomy and dependence on a server for a mobile user is very critical. A resource-scarce mobile system is better served by relying upon a server; however, frequent network disconnection, limited network bandwidth, and power restrictions argue for some degree of autonomy. As the environment changes, the application must adapt to the level of support required from stationary systems – the mobile-aware environment.

Overall, the main differentiating features between an MDAS and an MDBMS is the connection of servers and/or clients through a wireless environment and the devices used to access the data. However, both systems are intended to provide timely and reliable access to the globally shared data. As a result, many issues and solutions for an MDAS can be easily extracted from the MDBMS environment with added complexities introduced by the mobility and wireless connection of the MDAS environment.

2.4 Data Duplication in an MDAS

The communication limitations of an MDAS environment, i.e. restricted bandwidth, weak connections, and disconnection, may require that a portion of the data in some form be made readily available to the mobile unit. This data caching is used to increase the overall performance, and/or facilitate the continued operation of the mobile unit in case of disconnection. A large amount of related work has been done in the area of distributed file systems [17,25,26], distributed replication [10,16,23,27], and distributed/web caches [7,13,21].

The local autonomy requirement of an MDAS environment does not lend itself to the direct application of these works. There are two main objectives in maintaining duplicate data in a mobile or distributed environment:

- Data consistency: The duplicate copies of data should somehow be consistent within the system. Current schemes do not maintain transactional (ACID) consistency.
- Data access: When bandwidth limitations or disconnections occur, the impact on the availability of the data to the mobile user should be minimized as much as possible.

The scope of this paper addresses the ability to process data in a transactional sense in an MDAS environment – i.e., read and write operations are guaranteed to be consistent while connected to the system, and read-only operations are allowed during

disconnection.

2.5 Transactions

Data access in an MDAS is accomplished through transactions, where each transaction consists of read, write, and commit operations. A transaction is considered correct when the following properties (called ACID properties) hold for each individual transaction [12]:

- Atomicity. The operations of a transaction are considered to be atomic, either all the operations occur, or none occur.
- Consistency. The actions of a transaction taken as a group do not violate any integrity constraints of the database
- Isolation. Although transactions may execute concurrently, each transaction assumes that it is executing alone in the system. In other words, the system hides all intermediate results of a transaction from other concurrently executing transactions.
- Durability. Successful completion of a transaction (commit) should guarantee that the effects of the transaction survive failure.

2.5.1 Consistency Issues

Data duplication is used in mobile environments in order to mitigate the effects of the limited wireless connection of the mobile unit, where a weak connection or disconnection (either voluntary or involuntary) is considered a normal mode of operation. Under these conditions, a mobile unit accesses locally available data. The major drawback is the maintenance of the consistency of the local copy of data.

Several models were proposed in order to address the connectivity issues related to a wireless environment. These include weak and lazy consistency models that relax the consistency requirements of a system and allow the existence of inconsistent copies of data [2,22,27] – operations are performed on a conditional or delayed basis. These models eventually find discrepancies among the duplicate copies of data and attempt to reconcile the differences either by updating the information or semantically compensating for the discrepancy.

2.5.2 Data Replication

Replication schemes differ slightly from caching schemes in that the replicated data is accessible by other systems outside of the system on which the data resides. There are two types of general replication schemes, 1) primary/secondary copy (PSC) replication [21,23], and 2) voting-based replication schemes [13,27]. In PSC schemes, consistency is guaranteed by requiring that all write operations be performed on the primary copy of the data. This is done so that write inconsistencies do not occur. The write operations are propagated to the secondary copies of data. Read operations may access any primary or secondary copy of the data. Depending upon the PSC replication scheme, read operations may access data that is inconsistent with the primary copy. PSC replication does not work in an MDAS environment because write operations to replicated data do not reach the primary copy when disconnected. Therefore, write consistency is not guaranteed.

Voting based replications use multiple copies of the data in order to make a decision about the consistency of the data. Single operations of a transaction may be required to read and write to multiple sites (per operation). A voting process and/or

decision is made as to whether the read or write operation is successfully completed. In a distributed system, data consistency is guaranteed with voting based replications; however, they tend to be more expensive and require much more communication. Each read or write operation is sent to multiple locations, and vote is taken to determine which copy of the data is valid. In an MDAS environment, the local replica of data is not guaranteed to be consistent in the case of disconnection. While disconnected, the local system can not participate in the decision/voting process, and any changes to the local data may result in an inconsistency in the local copy.

2.5.3 Data Caching

Caching is an effective means of data duplication that is used to reduce the latency of read and write operations on data. Early research on the disconnected operation was done with file system based projects, e.g. the Coda file system [25]. When a disconnection occurs, a cache manager services all file system requests from the cache contents. Coda logs all transactions during periods of disconnection and replays the log upon reconnection in order to synchronize the two copies of data. Ficus [24], Little Work [14], Bayou [9], and Odyssey [26] are some projects that support the disconnect operation.

File or page based caching schemes use invalidation protocols in order to maintain the data consistency. The mobile unit and server keep track of cached data, and each time the data item changes, the server notifies the mobile unit. Alternatively, to reduce the amount of communication, the mobile unit can check for invalidated data only when accessing data. As with replication, the invalidation of data is not possible when disconnected, and thus consistency can not be guaranteed.

Web based caching is becoming more important as the information on the Internet continues its rapid expansion. The two most common forms of cache consistency mechanisms used on the Internet include time-to-live fields (TTL), and client polling [7]. TTL fields are an a priori estimate of the valid lifetime of an object. Client polling is a technique where clients periodically check the validity of a cached object. These caching mechanisms use the notion of weak consistency, and do not conform to the ACID properties.

There is currently no replication or cache-based scheme for mobile distributed systems that guarantees consistency. The disconnected operation of a mobile system makes consistency management extremely difficult. Some schemes try to use the concept of compensating transactions (or operations) in order to keep duplicated data consistent. However, compensating transactions are difficult to implement, and in some cases, a compensating transaction can not semantically undo the transaction (or operation). Furthermore, the local autonomy requirements of an MDAS render caching and replication schemes, even ones that use compensating operations, unusable in the MDAS environment.

In the next section, we address the bandwidth and local autonomy restrictions by using automated queued queries and caching of data in the form of a bundled query. The AQ2 uses a novel form of prefetching in order to pre-load data onto the mobile unit, such that the user can still access data in case of a disconnection. A bundled query is an object that consists of a query to the MDAS along with its associated data. The caching protocol for the bundled query is discussed.

3. Data Duplication Protocol

The use of duplicate copies of data has been extensively used and studied in order to provide availability and reduce latency. Maintaining the ACID property of transactions in the face of limited bandwidth, disconnection, and the local autonomy requirements of an MDAS environment introduce several limitations. In this section, we discuss how we address these issues.

3.1 Communication Protocol

As mentioned in the previous section, maintaining the ACID properties of a transaction with replicated data in an MDAS environment is very difficult. In our proposed scheme, we maintain strict read and write consistency while connected to the MDAS. While disconnected, since it is impossible to guarantee consistency, we allow the user to perform read-only queries on any data that is cached on the local unit. In such an instance, the user should be aware that any of the returned data from a query might contain stale information.

There are three levels of connectivity. In a strong connection, the mobile unit sends/receives all transactions and returns data directly to/from land-based, fixed sites for processing. When the communication link degrades to a weak connection, transactions are queued at the mobile unit and passed through the system according to the availability of bandwidth. Returned data is also queued at the fixed site. The queuing of operations during a weak connection allows a mobile unit to continue processing at the expense of increased latency.

While in the disconnect state, a user may perform read-only queries on any cached data. In such case, the consistency of the data is not guaranteed, i.e. the user may receive stale data. The user is limited to read-only access in order to ensure the validity of data in the MDAS. Any transaction that contains a write operation is queued and submitted to the system when the connection is reestablished. In addition, if a read-only access does not find the data locally, the query is queued and submitted later when the connection is established.

3.2 Cached Data - Bundled Queries (BUNQ)

Data that is cached on the mobile unit consists of a query and it's associated data. There are several reasons for using a BUNQ instead of page-based or file-based data. In a tightly coupled distributed system, it is possible to cache data at the local unit. However, in a multidatabase system, the structure of the data at the local systems will vary (structural differences – the information may be stored as structured data, unstructured data, web pages, files, objects, etc.). This variation of data at each local site makes it difficult to cache the data in a uniform manner. In addition, the autonomy requirement of the local system imposes further restrictions for caching data. In retaining local autonomy requirements, it may not be possible to determine the underlying structure of the data at the local site. For instance, to determine the pages of data accessed in a query from a local DBMS, knowledge regarding the tuples as well as the mapping to pages would be required. In order to provide this information, the local DBMS would have to be modified, directly violating autonomy requirements. Furthermore, the invalidation of the data at the local sites should be propagated to the mobile unit. This would also require major changes to the local systems, and hence, would violate local autonomy. Consequently, instead of caching individual data items from each

local source, the data set associated with a particular transaction is cached – a bundled query. By caching a BUNQ, the resolution of the structural differences is done at the MDAS level, while still maintaining the local autonomy requirements. The primary advantage is the ease and simplicity of implementation. This comes at the expense of retaining data in the cache at a very coarse grained level.

3.3 Prefetching

The proposed scheme uses a simple prefetching and replacement policy. The literature has addressed prefetching algorithms based upon user profiles and usage histories [17,21]. The limited power, storage, processing capability, and bandwidth of a mobile unit make the incorrect prefetch of data extremely expensive. A prefetching algorithm should maximize the data utilization. The idea is to prefetch enough data such that the user can still operate during a disconnection (albeit with relaxed consistency requirements), while minimizing the use of additional energy and bandwidth.

The data is prefetched by allowing the user to specify a particular read-only transaction as an automated queued query. An AQ2 has a relaxed requirement for consistency, which is defined by the user. The user sets a valid time threshold for each AQ2 when defining such a transaction. The mobile unit automatically submits the transaction to the MDAS when the threshold has expired – i.e. prefetches the data. The results are stored as a BUNQ. If the user requests the data in the AQ2 before the BUNQ is invalidated, the query is serviced from the local cache.

3.4 Replacement

The data in the cache consists of both automated queued queries and other user submitted read-only queries. The data in the cache is replaced based upon a least recently used (LRU) policy. The LRU policy has its advantages in that it is well understood and easy to implement. Moreover, other than some web-based caching algorithms, the LRU policy is the most widely used replacement policy in DBMS caches [7,11].

3.4.1 Invalidation

To maintain consistency between the copies of data residing on the fixed and mobile units, the data in the cache must be correctly invalidated when the main copy changes. To accomplish this, we propose using a parity-based signature for each BUNQ in the cache. After submitting a transaction to the MDAS, the fixed system generates a parity sequence that is appended to the BUNQ. When the user submits a transaction, if a corresponding BUNQ is present in the cache, the transaction (along with the parity code) is sent to the fixed node. The fixed system then performs the query and delays the transmission of the information back to the mobile unit until it generates and compares the two parity codes. If they are identical, only an acknowledgement is sent back to the mobile unit and the data is read locally from the cache. Otherwise, the resultant data, along with its new parity sequence, is returned and replaced in the cache. The old copy of the BUNQ is invalidated according to the LRU rules.

The use of a parity signature for invalidation incurs a small overhead. The parity sequence is generated for every user submitted read-only transaction, and once for each time quantum of the AQ2. The only additional overhead that the mobile unit

incurs is the added space required for storing the parity sequence. The number of bits required should be less than 100 bits per BUNQ, depending upon the probability of generating an erroneous parity sequence [19]. The fixed system will incur an additional cost of generating and comparing the parity codes. We believe that this is a very small price to pay for the reduction of communication between the mobile and fixed units. Figure 1 depicts the sequence of operations. In the next section, the simulation is presented, and results are discussed.

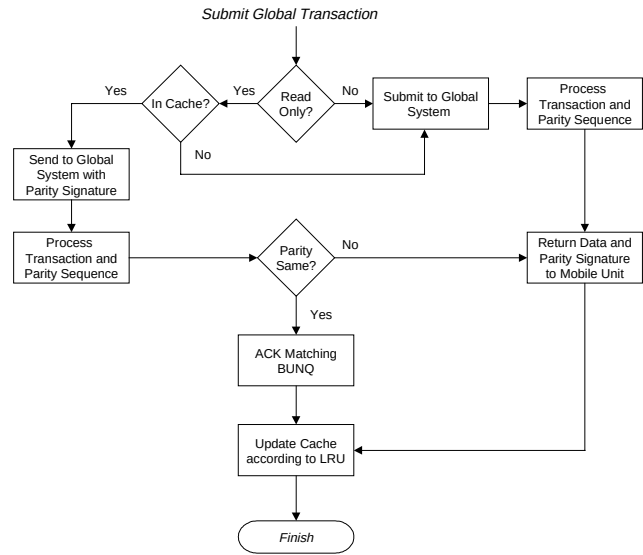


Figure 1: Invalidation Sequence Algorithm

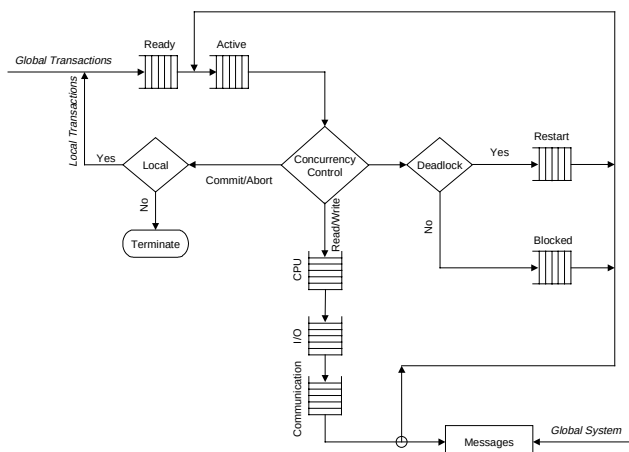
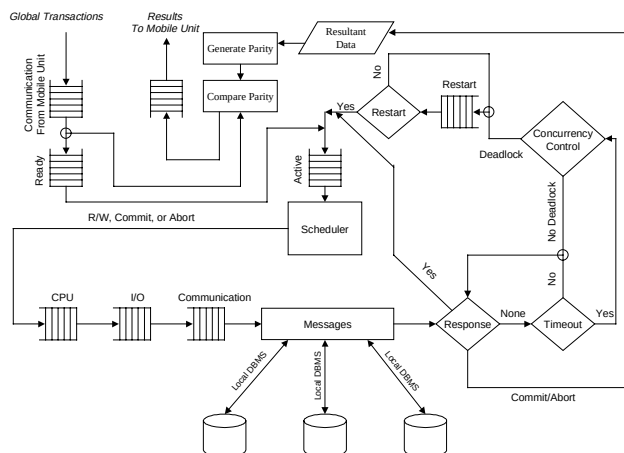
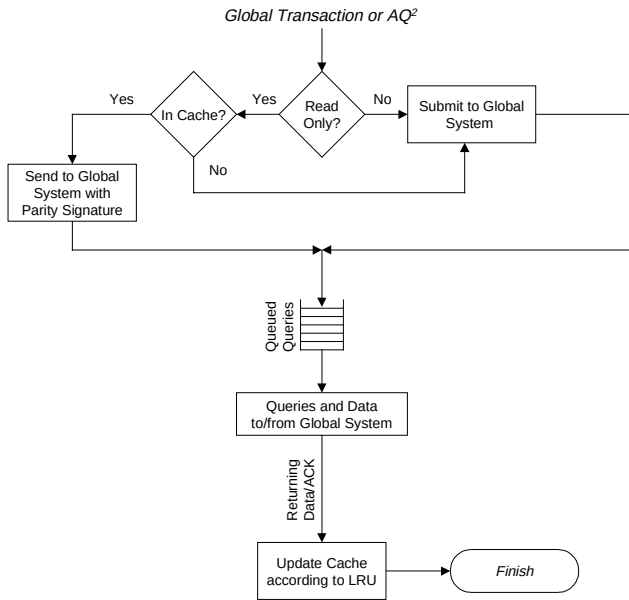
4. Simulation

The performance of the proposed algorithm (called p-caching) is evaluated through a simulator written in C++ using CSIM. The simulator measures performance in terms of global transaction throughput, response times, and CPU, disk I/O, and communication utilization. A comparison is made between a system with and without the use of AQ2 and BUNQ.

The MDAS consists of both local and global components, as well as mobile units. The local component is comprised of local DBMS systems, each performing local transactions outside the control of the MDAS. The global component consists of transactions executing under the control of the MDAS. The mobile units generate the actual global transactions. Figures 2, 3 and 4 depict the flow of operations in the mobile, global, and local units, respectively.

There are a fixed number of active global transactions present in the system at any given time. An active transaction is either in the active, CPU, I/O, communication, or restart queue. A global transaction received from a mobile unit enters a communication queue, and subsequently enters the active queue. Each operation of the transaction is scheduled and is communicated to the local system based upon the available bandwidth. When acknowledgements or a algorithm determines if the transaction should proceed, commit, or abort.

For read-only transactions following a global commit, a parity code is generated for the resultant data, and compared to the



parity code of the BUNQ. For a matching code, only an acknowledgement signal is sent back to the mobile unit. Otherwise, the data and the new parity code are sent back to the mobile unit. Transactions containing a write operation are placed directly in the ready queue. If a deadlock is detected, or an abort message is received from a local site, then the transaction is aborted at all sites and the global transaction is placed in the restart queue. After a specified time elapses, the transaction is again placed on the active queue.

At the local sites, there are a fixed number of active local transactions. The active transactions are comprised of both local transactions and global sub-transactions. The local system does not differentiate between the two types. An active transaction is defined as being in the active, CPU, I/O, communication, blocked, or restart queue. Local transactions enter the active queue and are subsequently scheduled by acquiring the necessary lock on a data item. If the lock is granted, the operation proceeds through the CPU and I/O queue, and is optionally communicated back to the global system. For global transactions, the acknowledgement is communicated back based upon the available communication bandwidth. A transaction goes into the restart queue if it is aborted, and subsequently will be restarted later. Upon a commit, a new local transaction is generated and placed in the ready queue, whereas global sub-transactions terminate.

4.1 System Parameters

The underlying global information sharing process is composed of ten local sites. The size of the local databases at each site can be varied, and has a direct effect on the overall performance of the system. The simulation is run for 5000 time units, and an average of 10 runs is taken for the values presented. The global workload consists of randomly generated global queries, spanning over a random number of sites. The queries are generated by a mobile unit and submitted to the global system. Once the transaction is successfully submitted, each operation of a sub-transaction (read, write, commit, or abort) may require data and/or acknowledgements to be sent to/from the local DBMS. The frequency of messages depends upon the quality of the network link. To determine the effectiveness of the proposed algorithm, several parameters are varied for different simulation runs. These parameters for the global system are given in Table 2, along with their default values.

Table 2: Global System Parameters

Global System Parameters	Default Value
Number of local sites in the system	10
Number of data items per local site	100
Service time for the CPU queue	0.005 sec
Service time for the IO queue	0.010 sec
Service time for each communicated message to the local site	0.100 sec
Number of messages per operation (read/write)	2

The local systems perform two different types of transactions – local and global. The global sub-transaction is submitted to the local DBMS and appears as a local transaction. The local transactions are generated at the local sites and consist of a

random number of read/write operations. The only difference between the two transactions is that a global sub-transaction will communicate with the global system, whereas the local transaction terminates upon a commit or an abort. The number of local transactions, which can be varied, affects the performance of the global system. Also, the local system may abort a transaction, global or local, at any time. If a global sub-transaction is aborted locally, it is communicated to the global system and the global transaction is aborted at all sites. The various parameters for the local system are given in Table 3 along with their default values. Both the global and local systems are modeled similar to the models used in [1,4].

Table 3: Local System Parameters

Local System Parameters	Default Value
Maximum number of local transactions per site. This number represents the local multiprogramming level.	10
Maximum number of write operations per transaction	8
Maximum number of read operations per transaction	8
Minimum number of write operations per transaction	1
Minimum number of read operations per transaction	1
Service time for the local CPU queue	0.005
Service time for the local IO queue	0.010
Service time for each communicated message to the MDAS	0.100
Number of messages per operation (read/write)	2

A collection of mobile units submits global queries (selected from a pool of 500 queries) to the MDAS. The connection between the mobile unit and the MDAS has a 50% probability of having a strong connection and a 30% probability of having a weak connection. There is a 20% probability of being disconnected. A strong connection has a communication service time of 0.1 to 0.3 seconds, while a weak connection has a service time range of 0.3 to 3 seconds. When a disconnection occurs, the mobile unit is disconnected for 30 to 120 seconds. Initially, one third of the pool consists of read-only queries, which are locally cached as a BUNQ. Additionally, 10% of the read-only queries are designated as an AQ2. The local cache is first checked for an existing BUNQ. If it is present, then the query is submitted along with the associated parity sequence. The MDAS returns either the data or the acknowledgement of a matching BUNQ. Subsequently, if signatures do not match, the mobile unit updates the cache according to the LRU scheme. Upon termination of a transaction, a new query is selected and submitted to the MDAS. The various parameters for the mobile units are given in Table 4, along with their default values.

4.2 Performance Results

The performance of the p-caching algorithm is evaluated relative to the throughput of global transactions, the percent of

transactions completed, the communication utilization at each local site, the hit ratio of the cache, and the effect of read-only data on the cache effectiveness. The simulation compares and contrasts the performance of an MDAS system with and without the proposed caching algorithm. Figure 5 shows the throughput results of the simulation. The throughput is very similar for the MDAS with and without caching, with the p-caching algorithm having a slightly higher peak throughput (1.37 vs. 1.58). This can be attributed to the read-only transactions (one third of the total transactions) completing locally on the mobile unit while disconnected, as well as the reduced communication cost when a BUNQ is acknowledged rather than transmitted. Initially, the throughput increases slowly due to the disconnection and relatively long period (30-120 seconds) that the mobile unit stays disconnected. As the number of concurrent global transactions increases, the global throughput decreases due to the increase in the number of conflicts and system thrashing. Both the caching and non-caching cases decrease at about the same rate due to the saturation of the communication link.

Table 4: Mobile Unit Parameters

Mobile Unit Parameters	Default Value
Maximum number of mobile units in the system. This number represents the global multiprogramming level.	10
Maximum number of operations that a global transaction contains.	8
Minimum number of operations that a global transaction contains.	1
Service time for the CPU queue	0.005 sec
Service time for the IO queue	0.010 sec
Service time for each communicated message to the global system for a strong connection	0.100-0.300 sec
Service time for each communicated message to the global system for a weak connection	0.300-3 sec
The service time for a disconnection	30-120 sec
Probability of a strong connection	0.50
Probability of a weak connection	0.30
Probability of a disconnection	0.20
Number of messages per operation (read/write)	2
Average size of each message	1,024 bytes
Size of the mobile unit cache	1 Megabyte
Probability that a query is read-only	1/3
Probability that a read-only query is submitted as an AQ ²	0.1

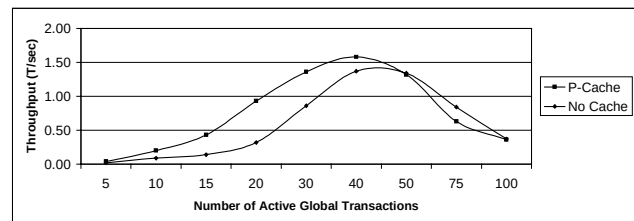


Figure 5: Comparison of the Global Throughput for Caching and Non-Caching Schemes

Figure 6 shows the percentage of completed transactions that the simulator measured. As expected, the p-caching algorithm shows a much higher percentage of transactions that complete. At peak throughput, where there are 40 active transactions, the p-caching algorithm shows an 85% completion rate, while without caching, there is only a 59% completion rate. The higher completion rate is due to the queuing of transactions during disconnection, and the accessing of cached data. Without a cache, the transactions are started and immediately aborted during disconnection. Furthermore, there is a reduction in communication from using bundled queries. A valid BUNQ only requires an acknowledgement signal, and does not incur the cost of transmitting the data. The percent of completed transactions is also much greater (48% vs. 20%) for a high number of active users (100 users) for the same reasons stated above.

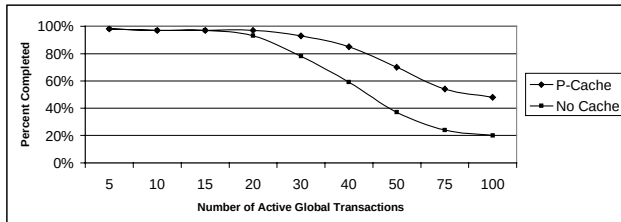


Figure 6: Comparison of the Percent of Global Transactions Completed

The communication utilization for the simulator is shown in Figure 7. For an active number of users of less than 30, the caching algorithm provided approximately a 10% decrease in the communication utilization. This decrease in communication is directly attributed to the read-only transactions being serviced locally. At peak throughput, both cache and non-cache based systems were using approximately 95% of the available bandwidth. As expected, this indicates that the communication in the system is the bottleneck.

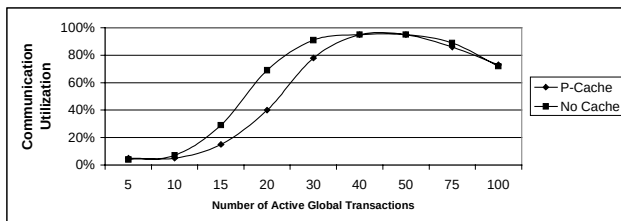


Figure 7: Comparison of the Communication Utilization at Each Local Site

The cache hit-ratio was measured by varying the cache size for 10, 40 (peak throughput), and 80 active users. The results in Figure 8 show that at peak throughput, the cache is somewhat ineffective for caches smaller than 256 Kbytes in size. After 1MB of cache, the curve flattens to a peak hit-ratio of approximately 34%. The write operations of the local and global transactions are continually invalidating data and therefore, the hit-ratio does not increase much after 1MB of cache. The cache hit-ratio at peak throughput was also similar for 10 and 80 users. This result is unexpected; however, we discovered that the invalidation of the data primarily comes from the local transactions.

Finally, the simulation was run by varying percentages of read-only transactions in the MDAS. The cache hit-ratio was measured at peak throughput (40 active users). As seen in Figure 9, the cache hit ratio increases as the percentage of the read-only transactions increases in the MDAS. The cache is not very effective until the size reaches 128 Kbytes in size, where the hit ratio is approximately 18% for accesses to entirely read-only data. For the 100% read-only case, the hit ratio approaches 100% with a 4MB cache. This is expected since the entire data set of 500 queries for the MDAS is approximately 5MB.

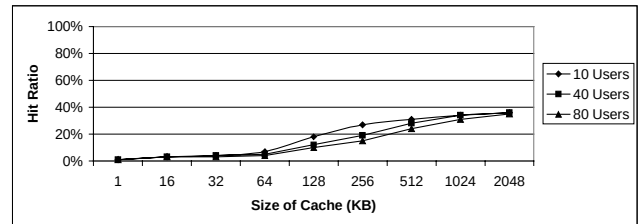


Figure 8: Cache Hit Ratio vs. Cache Size at Peak Throughput

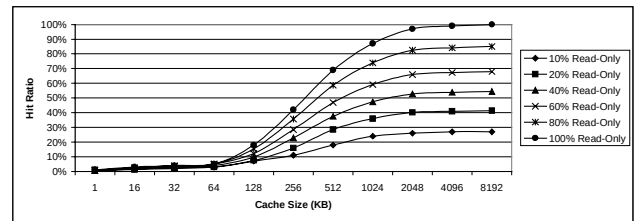


Figure 9: Cache Hit Ratio for Varying Amounts of Read-Only Data

5. Conclusion

The requirements of an "anytime, anywhere" computing environment motivate new concepts that effectively allow a user to access information in a timely and reliable manner. An overview of the characteristics of a new computing environment, the MDAS, and the requirements and issues of this environment were introduced and discussed. Recent technological advances allow users to access globally shared data through wireless connections via a diverse number of access devices. These devices differ in computing, energy, display, and network requirements.

Data duplication in the form of replication and caches is used to lessen the effects of weak communications or disconnection. The duplicated data at the mobile node allows the user to continue to work in the event of a weak connection or disconnection. With an additional copy of data available locally, the consistency of the data between a fixed node and a mobile node should be maintained – i.e. ensure the ACID properties of a transaction. Due to the local autonomy and disconnection of a mobile unit from the MDAS, maintaining write consistency is practically impossible.

A new caching scheme has been presented and evaluated in this paper. We use automated queued queries and bundled queries to address the limited bandwidth and local autonomy restrictions in an MDAS environment. To maintain data consistency between fixed and mobile units, read-only queries are cached as a bundled query and are validated using a simple

parity-checking scheme. Automated queued queries are used in order to prefetch data to the mobile unit.

A simulation was developed and the results were discussed. The results show that p-caching is effective in addressing the limited bandwidth and disconnection associated with the MDAS environment. In particular, we have shown that our algorithm decreases communication, and improves the percentage of completed transactions. Additionally, the overall throughput is increased due the decrease in communication.

5.1 Future Directions

The results we have demonstrated are promising. There are several areas of the work presented in this paper that we are investigating. The data set for the AQ2 is assumed to remain static for a minimum of at least 2 accesses and a maximum of 10 accesses. This is done in order to simulate slower changing data, and to see the effectiveness of the cache size. This simulated workload is not indicative of the many different workloads in an MDAS. The effect of different workloads and data sets of local sites should be investigated – i.e. a read-only environment, and various probabilities of read-only transactions.

Furthermore, the effect of the cache hit ratio on the MDAS environment (throughput, utilization, etc.) will be studied. This could be done by changing the cache-hit ratio to a raw probability that a read-only query is valid or invalid. The effect of the cache size could also be studied under these conditions. By setting the cache hit ratio instead of relying upon the actual invalidation of the data, we should be able roughly to simulate different workloads on the MDAS.

References

- [1] R. Agrawal, M. Carey, M. Livny, "Models for Studying Concurrency Control Performance: Alternatives and Implications," Proceedings of the ACM SIGMOD International Conference on Management of Data, 1985.
- [2] A. Adya, B. Liskov, "Lazy Consistency Using Loosely Synchronized Clocks," Proceedings of the ACM Symposium on Principles of Distributed Computing, 1997.
- [3] B.R. Badrinath, "Designing Distributed Algorithms for Mobile Computing Networks," Computer Communications, 19(4), 1996.
- [4] Y. Breitbart, A. Silberschatz, "Performance Evaluation of Two Multidatabase Transaction Management Algorithms," Computing Systems, 6(3), 1993.
- [5] M. Bright, A. Hurson, S. Pakzad, "A Taxonomy and Current Issues in Multidatabase Systems," IEEE Computer, 25(3):50-60, 1992.
- [6] M.W. Bright, A.R. Hurson, and S.H. Pakzad, H. Sarma, "The Summary Schemas Model – An Approach for Handling Multidatabases: Concept and Performance Analysis," Multidatabase Systems: An Advanced Solution for Global Information Sharing, IEEE Computer Society Press, 199-216, 1993.
- [7] P. Cao, C. Liu, "Maintaining Strong Cache Consistency in the World Wide Web," IEEE Transactions on Computers, 47(4), 1998.
- [8] K. Dash, A. Hurson, S. Phoha, C. Chehadah, "Summary Schemas Model: A Scheme for Handling Global Information Sharing," Proceedings of the International Conference on Intelligent Information Management Systems, pp. 47-51, 1994.
- [9] A. Demers, K. Pertersen, M. Spreitzer, D. Terry, M. Theier, B. Welch, "The Bayou Architecture: Support for Data Sharing among Mobile Users, IEEE Proceedings of the Workshop on Mobile Computing Systems and Applications," 1994.
- [10] M. Faiz, A. Zaslavsky, B. Srinivasan, "Revising Replication Strategies for Mobile Computing Environments," Proceedings of the ECOOP 95 Workshop in Mobility and Replication, 1995.
- [11] M.J. Franklin, M.J. Carey, M. Livny, "Transactional Client-Server Cache Consistency: Alternatives and Performance," ACM Transactions on Database Systems, 22(3), 1997.
- [12] J. Gray, A. Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993.
- [13] J. Gwertzman, M. Seltzer, "World-Wide Web Cache Consistency," Proceedings of the 1996 USENIX Technical Conference, 1996.
- [14] P. Honeyman, L. Huston, J. Rees, D. Bachmann, "The LITTLE WORK Project," Proceedings of the Third IEEE Workshop on Workstation Operating Systems, 1992.
- [15] A.D. Joseph, J.A. Tauber, M.F. Kaashoek, "Mobile Computing with the Rover Toolkit," IEEE Transactions on Computers: Special Issue on Mobile Computing, 1997.
- [16] D. Kottmann, "Serializing Operations into the Past and Future: A Paradigm for Disconnected Operations on Replicated Objects," Proceedings of the ECOOP 95 Workshop in Mobility and Replication, 1995.
- [17] H. Lei, D. Duchamp, "An Analytical Approach to File Prefetching," Proceedings of the 1997 USENIX Annual Technical Conference, 1997.
- [18] J.B. Lim, A.R. Hurson, "Heterogeneous Data Access in a Mobile Environment – Issues and Solutions," Advances in Computers, 1999.
- [19] J.J. Metzner, "Efficient Replicated Remote File Comparison," IEEE Transactions on Computers, 40(5), 1991.
- [20] B.B. Noble, A. Helal, "Mobile Computing and Databases: Anything New?," Sigmod Record, 24(4), 1995.
- [21] R. H. Patterson, G. A. Gibson, et al., "Informed Prefetching and Caching," Proceedings of the 15th Symposium on Operating System Principals, 1995.
- [22] E. Pitoura, B. Bhargava, "A Framework for Providing Consistent and Recoverable Agent-Based Access to Heterogeneous Mobile Databases," Sigmod 24(3), 1995.
- [23] E. Pitoura, "A Replication Schema to Support Weak Connectivity in Mobile Information Systems," Proceedings of the 7th International Conference on Database and Expert Systems Applications (DEXA), 1996.
- [24] P. Reiher, J. Heidemann, D. Ratner, G. Skinner, G. Popek, "Resolving File Conflicts in the Ficus File System," Proceedings of the USENIX Conference, 1994.
- [25] M. Satyanarayanan, J.J. Kistler, L.B. Mummert, "Experience with Disconnected Operation in a Mobile Computing Environment," Proceedings of the 1993 USENIX Symposium on Mobile and Location-Independent Computing, 1993.
- [26] M. Satyanarayanan, "Mobile Information Access," IEEE Personal Communications, 3(1), 1996.
- [27] Q. Wang, H. Paris, "Managing Replicated Data Using Referees," Proceedings of the ECOOP 95 Workshop in Mobility and Replication, 1995.