An Architecture for Consumer-Oriented Online Database Services *

Prasad Sistla[†], Ouri Wolfson[†], Son Dao [‡], Kailash Narayanan[†], Ramya Raj[†]

Abstract

In this paper we introduce an architecture for online database services oriented towards consumers. We identify two types of costs— access cost and communication cost. We demonstrate that dynamic allocation of data can minimize these costs. We do so by presenting efficient algorithms based on dynamic allocation; these algorithms optimize access and communication costs for various cost models, access patterns, and retrieval protocols.

1 Introduction

We are witnessing the emergence of a new form of computing/communication environment, called consumer (as opposed to business) computing. In such an environment, tens of millions of consumers will carry portable computers equipped with wireless data communicators that enable access to a large number of databases, digital libraries, and other online services. The potential market for this activity is estimated to be billions of dollars annually. For example, passengers will access airline and other carriers schedules, as well as weather information. Investors will access prices of financial instruments, sales people will access price and inventory data, route planning computers in cars will access traffic information, and callers will access location dependent data (e.g. where is the nearest doctor [1]).

There will be two types of charges incurred in this information-at-your-fingertips environment, namely access and communication charges. Access charges will be paid to the information provider, and communication charges will be paid to the network provider. For example, currently RAM Mobile Data Corp. charges on average $0.08 per data message to or from the mobile computer (the actual charge depends on the size of the message), and Data Broadcasting Corp. charges for providing the prices of financial instruments. Similarly, these two types of charges are incurred today in voice communication, when calling 900 numbers.

We believe that these charges will be the driving force that determines the overall system architecture, and the available access modes and protocols. Both types of charges can be optimized by the **dynamic allocation** principle. This principle states that when a customer reads a data-item more frequently than the data item is updated, then the amount of data transfers is minimized if the customer has a Subscription to receive all the changes to the data item; and vice versa, when a customer reads a data-item less frequently than the data item is updated, then the amount of data transfers is minimized if the customer Demands the data item from the online server at each access. In other words, a copy of the data item should be dynamically allocated and deallocated at the customer's computer, depending on the frequency of customer-reads compared to the frequency of data-item-writes.

The way in which this principle can be applied depends on the access and communication cost structures, the type of knowledge about the access pattern to the online service, and the retrieval protocols that are used. For example, if a data transfer on-Demand is twice as expensive as a data transfer by Subscription, then cost is optimized if a second copy of the data item is created when the number of reads is more than *half* the number of writes.

In this paper we demonstrate the application of the dynamic allocation principle for cost optimization. We do so in an architecture we propose, and for several types of access patterns, cost models, and retrieval protocols. Although the architecture and the cost models seem reasonable to us, we realize that currently

*this research was supported in part by NSF Grant Numbers IRI-9408750 and IRI-9224605, and AFOSR Grant Number F49620-93-1-0059.

[†]Electrical Engineering and Computer Science Department, University of Illinois, Chicago, Illinois 60607

[‡]Hughes Research Laboratories, Information Sciences Laboratory, Malibu, CA

the industry is in its infancy, and the way these factors will evolve depends on competitive market forces and technological advances that are hard to predict at this point in time. However, we believe that the principle can be applied in many other scenarios that will emerge.

We consider the following retrieval protocols. Assume that a customer accesses the prices of a portfolio of stocks. The access can use the Subscription protocol, in which the customer receives every update to a portfolio-stock price, or the on-Demand protocol, in which the customer requests the portfolio prices when needed. The on-Demand protocol may have a Cache-Invalidation option, in which, following a transfer of the data-item to the customer, the server informs the customer when the first update to the data-item occurs, without providing the new value. The Subscription protocol may be parameterized for divergence, allowing selective transmission of updates; for example, the customer may request a refresh only when his/her copy diverges from the most up-to-date copy by 3 updates, or 3%, or 3 minutes (see [2,3,4]).

We envision that the customer will use one of these retrieval protocols, and the one that optimizes the access cost depends on two factors: the access cost model, and the access pattern type. Furthermore, the user may be able to change the retrieval protocol as the access pattern changes. For example, the user may retrieve by Subscription between 9am and 10am, and on-Demand between 10am and 11am.

For optimizing the communication cost, the retrieval protocol may also be able to make use of the *broadcast option*. Specifically, the information provider may be able to choose whether to broadcast a particular update it receives, or to transmit it in a point-to-point fashion to one or more customers. Obviously, when broadcasting information that involves an access cost, the data will have to be encrypted, so that only the owners of a decryption key will be able to decipher it.

The cost model may be time-based or request-based, and the customer may have a choice between the two. In a time-based model, there is a flat fee per time period, e.g. a month. This enables the customer to receive all the price updates to his stock portfolio. In a request-based model the customer is charged per transfer of information, where the transfer may be a result of Subscription or Demand.

The optimal retrieval protocol also depends on the access-pattern. The access pattern to a database view (e.g. the stock portfolio) is the pattern of reads (by the customer) and updates (by the database server) of the view. The types of access patterns that we consider are deterministic, partially deterministic, and probabilistic. In a deterministic access-pattern, the customer/information-provider know the time of each read and each update (e.g. the customer may know that s/he reads the value of the database view every day at 5pm, and the view is updated at midnight). In a partially deterministic access pattern, for each time slot (e.g. between 9am and 10am) the number of reads and writes is known, although the exact time of each access is not known. In a probabilistic access pattern, expected number of reads and and the expected number of writes in each time slot are known. The customer and/or the information-provider define the access pattern, and input it to an algorithm that selects the optimal retrieval protocol for each slot.

Currently, we are building a system called Wireless-View, to be installed in mobile computers that access online services. It receives as parameters the expected access pattern and the cost model, and selects a retrieval protocol that optimizes the overall cost. In this paper we describe how the system makes this selection for the cost models, access patterns, and retrieval protocols discussed above. Some combinations of (cost-model, access-pattern-model, retrieval-protocol) have a straight-forward formula for computing the total cost, others are more challenging. For example, the (linear-time) allocation algorithm for partially deterministic patterns in the time based access-cost model was difficult to devise, particularly if the pattern is periodic and/or estimated (i.e. the number of reads (writes) in the next time-slot is estimated as the average of the number of reads (writes) in the last j slots). Things are further complicated by the fact that not all protocols for communication-cost optimization are compatible with protocols for access-cost optimization.

In summary, the main contributions of this paper are as follows.

- We demonstrate the application of the dynamic allocation principle for cost optimization in accessing online services.
- We introduce an architecture for online database

services in a consumer-oriented environment.

- We outline several cost models and factors affecting these costs.

- We present efficient algorithms for optimizing these costs for various access patterns and retrieval protocols.

Now we discuss relevant work. In a previous paper ([7]), we have also discussed optimal retrieval protocols, but that work was restricted to a particular scenario presented here, namely optimization of communication cost for probabilistic access patterns in the request based cost model. The work in [8,9] is also relevant to this paper but that work concentrates on broadcasting, which is not a central theme of our paper. The Wireless-View system discussed above was conceived in [6].

The rest of the paper is organized as follows. Section 2 discusses the architecture of a typical consumer-oriented online-services environment. In Sections 3 and 4 we discuss algorithms for access cost optimization. Section 3 concentrates on the time based cost model, and section 4 concentrates on the request based cost model. Section 5, discusses Communication Cost Optimization between the Server Agent and the Server.

2 Architecture

The environment consists of three independent functional entities, the database publisher, the network provider, and the customers. The database publisher provides services to wide geographic areas, e.g., the whole US. The customers use mobile/portable computers, thus the system consists of a set MC of active mobile computers and a server stationary computer SC that stores the online database. We assume that each mobile computer in the set MC is *connected* to SC, i.e., it can send and receive messages from the SC. The MC set varies over time since mobile computers can disconnect, turn-on and off. Although we refer to the computers in the set MC as mobile computers, some of them may be in a fixed location, connected by wirelines to the fixed network. What matters for the purpose of this paper is that the computers in the set MC access online databases and digital libraries using the architecture outlined in this section.

Each customer pays access charges to the database publisher, and the database publisher pays the network provider for the communication. Alternatively, the customer may pay communication charges directly to the network provider. In some cases the access charge is zero. For example, if the user is a salesperson that accesses the inventory information in the corporate database, access charge will probably be zero.

We consider retrieval and update of a database view, x. Here x is defined by a predicate, and it may contain one or more data items, or it may constitute the whole database. For example, x may be the IBM stock price, or the price of some collection of stocks, or it may be the set of records of AA flights out of Chicago on 12/12/93, or mailbox of the user in the server computer. In the latter case, a write is the addition of a message to the mailbox, and a read retrieves the unread messages.

We are concerned with reads that are issued at the mobile computers and writes that are issued at the server computer. These are the *relevant* requests.

The user retrieves information from the view x either by Subscription, or on-Demand. A Subscription requests that all updates to x be transmitted to the user's mobile computer. We also discuss the case of selective Subscriptions, where the update is transmitted only when some threshold is exceeded (e.g. the price of the stock exceeds 15, or the view in the mobile computer is more than 10 minutes old).

In order to access the database a user has to install in his/her mobile computer a software module provided by the database publisher. This software module is called the Server Agent (SA); it processes the requests of the Client Agent (CA), i.e. the user or his/her software. The CA receives database information from the SA, and in turn, the SA receives information from the server.

This architecture illustrated in Figure 1.

The SA provides to the CA the following services for each view x.

- *subscribe*: Unsolicitedly, transfer to the CA each update of x, or, in the case of divergence caching, selective updates. The transfer is received and processed by the CA while the user may be using his/her consumer computer for various other purposes.

- *cancel subscription*: The CA indicates to the SA

The Architecture

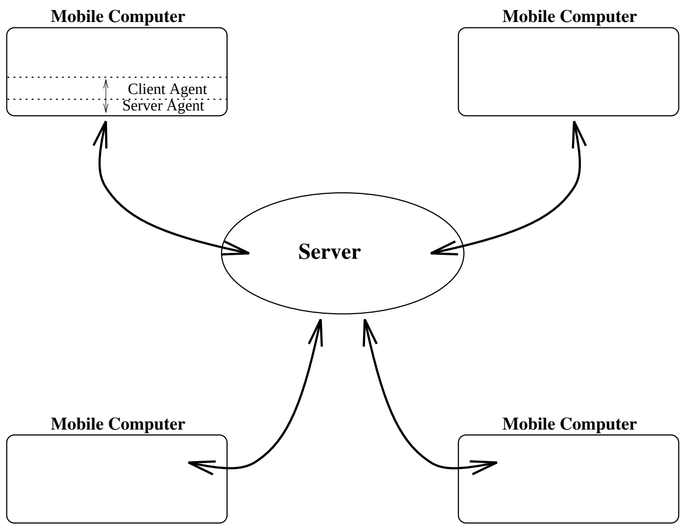

Figure 1: Architecture

that the CA is not to receive further updates to the view x.

- *read* : Transfer to the CA the current contents of x. This request may be asynchronous in the sense that the Client does not wait for x, but proceeds with performing other functions.
- *invalidate*: The SA indicates to the CA that there was an update of the x, without providing the new contents.

Access Cost

The interaction between the SA and the CA incurs access cost. We will consider two access-cost models. The first, denoted ACM1, is *request-based* in the following sense. If the user has a Subscription to receive all the updates to x, then, for each transfer of information from the database to the user, the user is charged a *subscription-access-cost*. This cost is denoted *sac*. On the other hand, if the user submits a query to receive the current copy of x, then the cost of the service is *dac* (*demand-access-cost*), where $dac \geq sac$. We assume that the invalidate and cancel-subscription requests do not incur an access cost. This is the cost model of newspapers and magazines, where the cost per copy is higher at the news-stand than for subscription-delivery.

The second access cost model, denoted ACM2, is *time-based* in the following sense. A Subscription enables the user to receive all writes to the view for a flat fee *ff* per time unit (say a day). In other words, the cost per time unit is fixed, regardless of the number of writes propagated to the user. There is also a Subscription initiation fee, *if*, which is greater or equal to zero. This is the cost model for telephone (unlimited number of local calls) or cable TV service. In this cost model the user can also submit a query to receive the current copy of x, if s/he does not have a Subscription. Then the cost of the service is *dac*. In this cost model the invalidate request is not being used, and the cancel Subscription request carries a zero access-cost.

Communication Cost

We assume that the network provider enables both, point-to-point and broadcast communication between the server and a mobile computer. For example, the broadcast facility may be used for transmitting invalidation messages, and the point-to-point facility may be used for data (this is the scheme used in [5]).

The communication between the SA and the Server incurs *communication cost* as follows. A point-to-point transmission of a page (or less) from the Server to the Client costs cc (charged by the network provider), and the cost of a request (to read) sent from the Client to the server is ω. Note that ω may be higher than cc since uplink communication may be more expensive than downlink communication. The broadcast of a page from the Server to an arbitrary number of Clients costs B.

Retrieval Protocols

We will consider the cost of two basic retrieval-protocols for reading data from the view x. Each one of these protocols can be used for CA's interface to retrieve x-data from the SA, and for SA's interface to retrieve x-data from the server. The objective of each interface is to optimize cost.

The architecture involving access and communication costs is depicted in Figure 2 .

The Retrieval Protocols that we consider are Demand, denoted D, and Subscription, denoted S. If the CA retrieves x-data from the SA using Demand, it means that the CA requests the current value of x for each read issued by the user. In other words, the CA does not keep an up-to-date copy of x. Similarly, if the SA retrieves x-data from the server using Demand, it means that the SA requests the current value of x whenever needed.

As a feature of the Demand protocol the CA (SA) may request *Cache invalidation*. This means that the SA (server) sends an indication to the CA (SA) whenever the view x is updated. Since this indication is cost-free, the feature enables the Client to pay only for reads that are immediately preceded by a write. Another way to implement Cache invalidation is to charge the user only for reads that are presented when the value of the view x is different from the one that existed when the immediately preceding read was presented. The Cache invalidation feature may or may not be available.

If the CA retrieves x-data from the SA using Subscription, it means that the CA requests to automatically receive all the updates to the view x. In other words, the CA keeps an up-to-date copy of x. Similarly, if the SA retrieves x-data from the server using Subscription, it means that the SA requests to automatically receive all the updates to the view x. The Subscription access protocol may be parameterized for divergence (see [3,4]). Specifically, the Subscription protocol receives a parameter that enables a selective transmission of updates. For example, the CA may receive one in every 3 updates to the view x. This will be elaborated upon in section 6.

Access Patterns

For a given Retrieval Protocol, the access and communication costs depend on the cost model, which we discussed above, and on the type of access pattern, which we discuss next. The user can provide access-pattern information in one of several types. One is *deterministic*, in which the user specifies a set of time stamped relevant requests. For example, the user specifies that there is a write of x every hour on the hour, and reads at 4:05pm, 4:30pm, 4:50pm and 6:05pm. Clearly, for each one of the cost models and for each protocol the cost of a given pattern can be computed in a straight-forward manner. Unfortunately, Access Pattern information is rarely available in this form.

A second type of access pattern is *partially deterministic*, in which the user specifies the number of requests for each time unit. For example the user specifies that between 9 and 10 there are 3 reads and 4 writes, between 10 and 11 there are 2 reads and 5 writes, etc. Determinacy is only partial, because the exact time of each request is not known.

A special case of partial determinism is when the number of requests in each time unit, instead of being explicit is estimated as a function of the previous n time units. For example, the number of reads in any particular hour is is estimated to be the average of the number of reads in each of the last five hours.

A third type of access pattern is *probabilistic*, in which the user specifies the expected number of requests for each time unit. We assume that the numbers of reads and writes in each time unit are random variables; we assume that any pair of these random variables are independent. Thus, this model is similar to the partially deterministic one, except that the parameters are expected in the probabilistic sense, rather than exact ones.

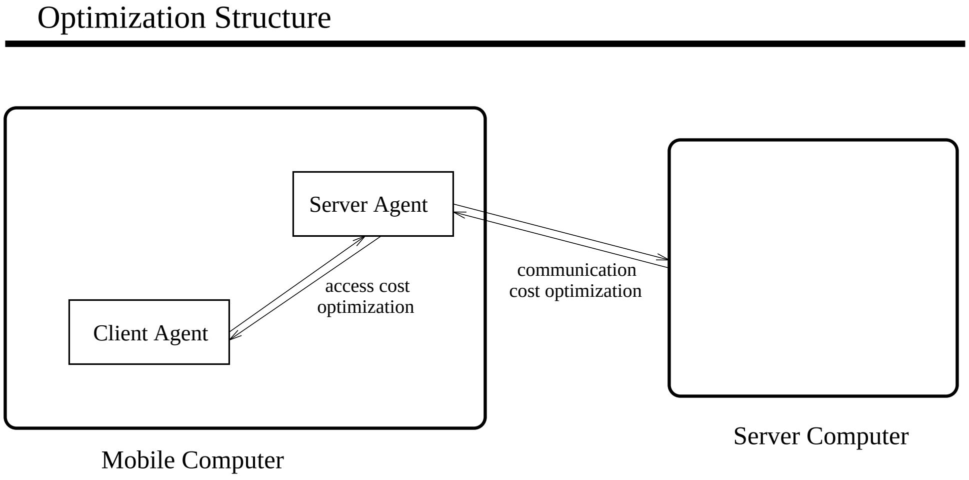

Figure 2: Optimization Structure

Each one of the above access pattern types can be specified for a period such as a day, a week, or a month. It can be specified that the Access Pattern is repeated each period.

The database publisher provides the write pattern to each consumer (user) in order for the consumer to be able to optimize the access cost, and the consumer provides the read pattern to the database publisher, so that the latter can optimize the communication cost. Thus, both the Client and the Server know a priori the type of access pattern, and its parameters. Obviously, we assume that the read and write access patterns are compatible, i.e. if the consumer provides a probablistic read pattern, then the database publisher also provides the write pattern in a probablistic format, so that both, the consumer and the database publisher, can arrive at a common probablistic access pattern.

3 Access Cost Optimization in the Time-Based Cost Model

We consider the issue of access cost optimization between the Client Agent and the Server Agent in the mobile computer. We have two Cost Models, three Retrieval protocols and a variety of Access Patterns. Thus we have a number of possible configurations. In this section we consider the Time based cost model and describe algorithms for access cost optimization. In the first subsection, we consider the *Subscription* and *Demand* Retrieval Protocols, i.e. we determine for each slot in the pattern which of the two protocols to use. In the second subsection, we consider estimated partially deterministic pattern and provide an algorithm for access cost optimization. In the third subsection, we consider periodic partially deterministic patterns. In the fourth subsection, we consider the Retrieval Protocol *Demand with Cache invalidation*.

3.1 Partially Deterministic Pattern, Time Based Cost Model

In this subsection we present an algorithm, called *Opt*, for selecting the optimal retrieval protocols for a given partially deterministic access pattern. The objective is to optimize access cost in the time based cost model. It is assumed that we can subscribe only at predefined times, e.g. midnight. Thus, we can divide the entire pattern into a number n of time slots.

Given a partially deterministic pattern, the objective of the *Opt* algorithm is to produce an *allocation pattern* i.e. assign Subscription or Demand to each time slot in the pattern. This determines the Retrieval Protocol used for the slot. We will show that for the given pattern, the allocation pattern gives the minimum total access cost.

The algorithm *Opt* operates as follows. It examines the slots one after another. After examining each slot, it either immediately decides a Retrieval Protocol for the slot, or it keeps the decision for the slot pending until it examines sufficient number of future slots. Thus, at any point in the execution of the algorithm,

$$\underbrace{SDD\cdots S}_{Decided\ Slots}\ \underbrace{\overbrace{-+\cdots+}^{cs}}_{Undecided\ Slots}\ \underbrace{2r\ 3w \mid 5r\ 6w \mid \cdots}_{Unexamined\ Slots}$$

Figure 3: Classification Of Slots

the sequence of time slots can be divided in to three types, (1) the sequence of *Decided* slots which occur in the beginning, followed by (2) the sequence of *Undecided* slots, i.e. slots that have been examined but for which a decision is pending, followed by (3) the sequence of *Unexamined* slots. The Decided slots are the slots for which Subscription or Demand protocol has been assigned. The algorithm uses two variables *status* and FUS (first undecided slot). At any time the *status* variable denotes the retrieval protocol assigned to the last decided slot, while FUS gives the index of the first undecided slot. The algorithm also uses a precomputed value df_i for each slot i. Here, df_i is the demand fee for the i^{th} slot and is computed as the product of the number of reads in the i^{th} slot and the demand access cost for a single read request. After the termination of the algorithm, the array variable *decision* contains the decisions that have been taken for each of the slots. Figure 3 shows the classification of the different slots.

Initially, the *status* variable is assigned Demand. The algorithm *Opt* scans the slots one by one, starting from the first. For each time slot i, it computes $(df_i - ff)$, where df_i is the demand fee of the i^{th} slot as described above, and ff is the flat fee per time unit. Assume that the value of the status variable is Demand. If the demand fee (df_i) is less than or equal to flat fee (ff), i.e. $(df_i - ff) \leq 0$, the algorithm *Opt* allocates Demand to the current (i^{th}) slot. If the demand fee is greater than the flat fee, i.e. $(df_i - ff) > 0$, it means that Subscription is cheaper than Demand. However, we cannot assign Subscription whenever this condition is satisfied, because Subscription also involves the payment of the initiation fee (if). So, we check whether the demand fee (df_i) is greater than the flat fee (ff) by if, i.e. $df_i - ff \geq if$. However, if the condition is not satisfied for a single slot, we cannot assign Demand to that slot. This is because the cumulative sum(cs) of $(df_i - ff)$ for a sequence of consecutive slots may satisfy the condition, i.e. $\sum(df_i - ff) \geq if$, although for each individual slot i the difference $(df_i - ff)$ may be positive or negative. So, algorithm *Opt* proceeds to the next slot and computes the cumulative sum (cs), $\sum(df_i - ff)$.

In summary, when the value of the *status* variable is Demand (i.e. the retrieval protocol assigned to the last decided slot is Demand), the Demand protocol is assigned to all the Undecided slots, i.e. from the First Undecided Slot to the current (i^{th}) slot, whenever $cs \leq 0$. If $cs \geq if$, then Subscription is assigned to all the Undecided slots and the status variable becomes Subscription. If neither of the above conditions is satisfied, the algorithm proceeds to the next slot and computes the cumulative sum and repeats the above operations.

If the status is Subscription, it means that the initiation fee has been paid, and we can allocate Subscription to all the Undecided slots whenever the cumulative sum (cs), $\sum df_i - ff \geq 0$. If the $cs < 0$, it means that Demand is cheaper for the Undecided slots. However, we cannot choose Demand if $\mid cs \mid < if$, because if we have Subscription in an Unexamined slot, we would be paying more in switching back to Subscription. Thus the algorithm *Opt* checks if $\mid cs \mid \geq if$, i.e. it checks if initiation fee is recoverable. If so, the algorithm chooses Demand for the Undecided slots.

In summary, when the status is Subscription, the Subscription protocol is allocated to all the Undecided slots whenever $cs \geq 0$. If $cs < 0$ and $\mid cs \mid \geq if$, then Demand is assigned to all the Undecided slots and the status becomes Demand. If neither of the above conditions is satisfied, the algorithm *Opt* proceeds to the next slot and computes the cs and repeats the above operations.

If we are unable to arrive at a decision based on the cumulative sum and we reach the end of the pattern, there are two possible cases, $cs < 0$ and $0 < cs < if$. In either case, the algorithm *Opt* chooses Demand for all the Undecided slots. This is justified because, if $cs < 0$, Demand is cheaper and if $0 < cs < if$, it means that the initiation fee is not recoverable.

A formal presentation of the algorithm is given in the appendix.

3.2 Estimated Partially Deterministic Pattern

This subsection considers estimated partially deterministic patterns in which the number of reads (writes) in the next time slot is estimated to be the floor average of the numbers of reads (writes) in the last j time slots. The allocation algorithm is on-line and at the beginning of each time slot, it computes the estimated value of the number of reads for the next one or more time slots, and uses these estimated values for selecting Demand or Subscription for the next time slot.

The *Opt* algorithm for the partially deterministic patterns given in the previous section can be adapted to the above setting by combining it with the computation of the estimated values, as follows. At run time, at the end of each time slot, the estimated number of reads for the next time slot is computed. Then, algorithm *Opt* is used to check if a decision can be reached; if so, we use the corresponding selection. Otherwise, i.e. if the slot is Undecided, we compute the estimated values for the time slot after the next, and repeat the above procedure until a decision about the next time slot (which is the First Undecided Slot) can be made. It can be shown that the estimated values will eventually converge, and after this the cumulative sum (i.e. the value of the variable cs) of the algorithm *Opt* (see section 3.1) will monotonically increase or decrease; at that time, we choose Subscription for the next time slot if the cumulative sum monotonically increases, otherwise we choose Demand for the next time slot. It can be shown that the estimated values converge within c iterations where c is a function of j and the maximum number of reads in any of the last j time slots.

We have a simpler and more efficient method for the case $j = 2$, but we omit this discussion for space considerations.

3.3 Periodic Partially Deterministic Pattern

Another special case of partial determinism is the case when the access pattern is periodic. Here, the numbers of reads and writes are known for each slot in one period of the pattern. The same pattern repeats periodically, infinitely.

$$D \cdots DS \cdots S \underbrace{D \cdots D}_{2} || \underbrace{D \cdots D}_{1} S \cdots D$$

Figure 4: Non-trivial Case

$$DDD \underbrace{+ \cdots - || - + - + \cdots -}_{Undecided\ Slots} || \cdots$$

$$\Downarrow$$

$$DDDS \cdots S||SSSS \cdots S||S \cdots$$

Figure 5: FUS in the 1^{st} period

The allocation pattern obtained for the first period cannot be directly assigned to all the other periods for the following reason. When we consider two adjacent periods we may have the case illustrated in Figure 4.

There, the segments 1 and 2 are in between two Subscription slots. It is possible that the cumulative sum of all slots in 1 and 2 is greater than $-if$. If this is the case, then all slots in 1 and 2 should be allocated Subscription. For this reason we modify the algorithm as follows and apply to the first two periods. The *Opt* algorithm is modified so that if either of the inner *Repeat* loops terminate because the end of the second period is reached, then for all the Undecided slots from First Undecided Slot to the end of the second period, the allocation pattern is chosen as follows. If the First Undecided Slot is in the first period, then we choose Subscription for all of the Undecided slots. This is illustrated in Figure 5. Otherwise, i.e., if the First Undecided Slot is in the second period, then for each i^{th} slot ($i \geq FUS$) in the second period, we choose the allocation pattern of the corresponding slots in the first period. This is illustrated in Figure 6.

For all other periods, following the second period, we repeat the allocation pattern of the second period.

The above allocation method can be shown to produce an optimal allocation pattern. The proof of correctness of this method is omitted due to lack of space.

$$SSD\cdots D\underbrace{S}_{j1}\cdots\underbrace{S}_{k}\|SSD\cdots D\overbrace{\underbrace{+}_{2j1}\cdots\underbrace{+}_{2k}}^{Undecided\ Slots}$$

$$\Downarrow$$

$$SSD\cdots SDS\cdots S\|SSD\cdots DS\cdots S$$

Figure 6: FUS in the 2^{nd} period

3.4 Demand with Cache invalidation

In this subsection we consider the Retrieval Protocols Subscription and Demand with Cache invalidation. We assume that the pattern is partially deterministic. The objective is to optimize access cost in the time based cost model. In the Retrieval Protocol Demand-with-Cache-invalidation, the Client has to pay only for the reads that are immediately preceded by a write, i.e. the critical reads. The expected number of critical reads n_c, assuming that there are n_r reads and n_w writes in a time slot and assuming that they are uniformly distributed, is given as follows.

$$n_c = \frac{n_r \cdot n_w}{n_r + n_w}$$

Then the expected demand fee for a slot is $n_c \cdot dac$. With this revised demand fee, the algorithm *Opt* can be applied verbatim for access cost optimization.

The algorithms that optimize access cost for the estimated and periodic patterns can also be applied verbatim using the so revised demand fee.

4 Access Cost Optimization in the Request-Based Cost Model

In this section, we consider the combinations (Access Pattern, Retrieval Protocol) in the request based cost model, and we develop algorithms for access cost optimization.

In the request based cost model, whenever the user submits a query to receive the current copy of the view x, then the cost of the service is demand-access-cost (dac). If the user has a subscription to receive all the updates to x, then for each transfer of information from the server to the user, the user is charged a subscription-access-cost (sac).

First we consider the case in which the possible choices of retrieval protocols are Subscription and Demand, i.e. the Cache-Invalidation option is unavailable. As before, there are three types of access patterns, i.e. deterministic, partially deterministic and probabilistic.

For each type of access pattern, the algorithm for access cost optimization is trivial. For each time slot i we need to compare the Subscription fee $s_i = n_w \cdot sac$ and the demand fee $d_i = n_r \cdot dac$, and assign the protocol that incurs a lower cost.

Now, what exactly are n_r and n_w in the above formula? If the access pattern is deterministic, the Client needs to pay only for those reads that follow a write. Thus n_r is the number of reads that immediately follow a write, and n_w is the total number of writes in the slot.

If the access pattern is partially deterministic, we are given the number of reads (n_r) and writes (n_w) in every time slot.

If the access pattern is probabilistic, n_r is the expected number of reads in time unit, and n_w is the expected number of writes in unit time. Obviously, in the probabilistic case, the cost optimized is the expected cost rather than the exact one.

Now assume that the cache invalidation option is available. As before, for each type of access pattern we compare the Subscription cost, $s_i = n_w \cdot sac$ with the Demand cost, $d_i = n_c \cdot dac$, and choose the protocol that gives the minimal cost for that slot. The only difference is that we use n_c (rather than n_r). This is the number of critical reads, i.e. reads that immediately follow a write in the slot.

5 Communication Cost Optimization

In this section we consider the issue of communication cost optimization between the Server and the Server Agent in the mobile computer. Again, there are three retrieval protocols that could be used by the Server Agent, i.e. Subscription (S), Demand (D), and Demand with Cache invalidation (D_CI).

We assume that the Retrieval Protocol between the Client Agent and the Server Agent (called the Access Protocol) has been fixed. The reason is that the consumer can, independently of the database publisher, select a retrieval protocol for access cost optimization. This imposes a constraint on the possible choices of Retrieval Protocols between the Server Agent and the Server (called the Communication Protocol). In order to see this consider the following case. Let the retrieval protocol between the Server Agent and the Client be Subscription. Then all the updates have to be propagated by the Server Agent to the Client. It is clearly evident that Demand cannot be used as the Retrieval Protocol between the Server Agent and the Server. Thus in this case, the only available retrieval protocol between the Server Agent and the Server is Subscription. The table below gives the possible choices of Retrieval Protocols between the Server Agent and the Server, for a given protocol between the Client Agent and the Server Agent.

Access Protocol	Communication Protocol
Subscription	*Subscription*
Demand	*Subscription*
Demand	*Demand*
Demand	*D _ CI*
D _ CI	*D _ CI*
D _ CI	*Subscription*

For the rest of this section we assume that access pattern is partially deterministic, but our approach can be applied to other types of access-patterns. Now we select for every mobile computer and for every time slot the retrieval protocol that minimizes the point-to-point (*ptp*) transmission cost, from the available protocol choices. If the broadcast mode is unavailable then our protocol selection process ends here. Thus, assume for the rest of this section that broadcast is available. We will determine whether or not the broadcast mode can further reduce the cost of the optimal *ptp* protocol. As mentioned in section 2, the broadcast-cost of a single write/page is B, and at this cost the page can be sent to all the active mobile computers.

Assume that the number of mobile computers that are active, i.e. turned on, at any point in time is known a priori, and the access pattern of each one of them is also known. This is the case, for example, in a company that has a fixed number of salespersons accessing the inventory database. Then we compute the following inequality for each time slot.

$$B \cdot n_w \leq k1 \cdot n_w \cdot cc + \sum_{i=1}^{k2} n_{ri}(cc+\omega) + \sum_{i=1}^{k3} n_{ci}(2 \cdot cc+\omega)$$

In the above formula $k1$ is the number of mobile computers with Subscription as the optimal *ptp* retrieval protocol, and n_w is the number of writes in the time slot. Observe that n_w is identical in the access patterns of all mobile computers, since it is the number of writes at the server, and is independent of a particular mobile computer. $k2$ is the number of mobile computers with Demand as the optimal *ptp* protocol; n_{ri} is the number of reads of the i^{th} such mobile computer. $k3$ is the number of mobile computers with Demand-with-Cache-Invalidation as the optimal *ptp* protocol; n_{ci} is the number of critical reads (for the method of computing the number of critical reads, see subsection 3.4) of the i^{th} such mobile computer.
If the above inequality is satisfied, then the cost of broadcasting each write is less expensive than the total cost of using the optimal *ptp* communication protocol for each mobile computer. Then each SA will use the Subscription communication protocol (which, as can be seen from the previous table, is compatible with every access protocol), and the data will be broadcasted during this time slot. Otherwise, i.e. if the inequality is not satisfied, each mobile computer will use in the slot the optimal *ptp* protocol selected in the first stage.

Now, whether or not the number of active mobile computers is known a priori, consider a time slot in which Subscription-with-broadcast is not the optimal protocol. At run time, the broadcast mode can still be used to reduce communication cost, as follows. When a write is received at the server, the database server computes k, the the number of mobile computers that have a Subscription to receive the write in a *ptp* mode, or, have to receive an invalidation *ptp* message as a result of the write. Then the Server determines whether the following inequality is satisfied.

$$B \leq k \cdot cc$$

cc is the cost of *ptp* transmission of a single page. If the above inequality holds, the write is broadcasted. Otherwise the write and 'invalidate' messages are transmitted point-to-point.

References

[1] Tomasz Imielinski and B.R.Badrinath. Mobile Wireless Computing : Challenges in Data Management. *Communications of the ACM*, 37:(10), Oct. 1994

[2] R. Alonso, D. Barbara, and H. Garcia-Molina. Quasi-copies: Efficient data sharing for information retrieval systems. In *Proc. of EDBT '88, LNCS 303*. Springer Verlag, 1988.

[3] R. Alonso, D. Barbara, and H. Garcia-Molina. Data caching issues in an information retrieval system. *ACM Trans. Database Syst.*, 15(3):359–384, 1990.

[4] Y. Huang, R. Sloan, O. Wolfson. Divergence Caching in Client-Server Architectures. *Proceedings of the third International Conference on Parallel and Distributed Information Systems (PDIS),* Austin, TX, Sept. 1994, pp. 131-139.

[5] Daniel Barbara, Tomasz Imielinski. Sleepers and Workaholics : Caching Strategies in Mobile Environments. *ACM Sigmod*, 1994.

[6] O. Wolfson. Data Allocation in Mobile Computing: A Project Description. *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems,* Princeton, NJ, Oct. 1993, pp. 89-94.

[7] Y. Huang, P. Sistla, O. Wolfson. Data Replication for Mobile Computers. *Proceedings of the ACM-SIGMOD 1994, International Conference on Management of Data,* Minneapolis, MN, May 1994, pp. 13-24.

[8] Tzi-cker Chiueh. Scheduling for Broadcast-based File Systems. *Mobidata Workshop,* Nov. 1994.

[9] S. Acharya, R. Alonso, M. Franklin, S. Zdonik. Broadcast Disks : Data Management For Asymmetric Communication Environments. *Proceedings of the ACM-SIGMOD 1995, International Conference on Management of Data.*

The Algorithm *Opt*

```
{
i:=1
last:=n
status:=Demand
while i ≤ last
    { cs := 0
    FUS:=i
    If status=Demand
        { repeat
            cs := cs + (df_i − ff)
            i := i + 1
        until ((cs ≤ 0) ⋁ (cs ≥ if) ⋁ (i > last))
        if (cs ≥ if)
        status:=Subscription
        }
    else
        { repeat
            cs := cs + (df_i − ff)
            i := i + 1
        until ((cs ≥ 0) ⋁ (cs ≥ −if) ⋁ (i > last))
        if ((cs ≥ −if) ⋁ (i > last))
        status:=Demand
        }
    for j = FUS up to i − 1
    decision[j]=status
    }
}
```