Semantic-Based Delivery of OLAP Summary Tables in Wireless Environments*

Mohamed A. Sharaf
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
msharaf@cs.pitt.edu

Panos K. Chrysanthis
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
panos@cs.pitt.edu

ABSTRACT

With the rapid growth in mobile and wireless technologies and the availability, pervasiveness and cost effectiveness of wireless networks, mobile computers are quickly becoming the normal front-end devices for accessing enterprise data. In this paper, we are addressing the issue of efficient delivery of business decision support data in the form of summary tables to mobile clients equipped with OLAP front-end tools. Towards this, we propose a new on-demand scheduling algorithm, called *SBS*, that exploits both the derivation semantics among OLAP summary tables and the mobile clients' capabilities of executing simple SQL queries. It maximizes the aggregated data sharing between clients and reduces the broadcast length compared to the already existing techniques. The degree of aggregation can be tuned to control the tradeoff between access time and energy consumption. Further, the proposed scheme adapts well to different request rates, access patterns and data distributions. The algorithm effectiveness with respect to access time and power consumption is evaluated using simulation.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Distributed databases, Query processing*; H.4.2 [**Information Systems Applications**]: Types of Systems —*Decision support*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms, Design, Experimentation, Performance, Theory

Keywords

Broadcast Scheduling, Broadcast Pull, Mobile Computing

*This work is supported in part by NSF award ANI-0123705 and by National Center for Disease Control.

CIKM'02, November 4–9, 2002, McLean, Virginia, USA.

1. INTRODUCTION

With the rapid growth in wireless technologies and the cost effectiveness in deploying wireless networks, wireless devices are quickly becoming alternative platforms for accessing enterprise data. This, combined with the increased popularity of palmtop and hand-held computers as well as the availability of light yet powerful laptop computers, mobile computers will become the normal front-end devices hosting sophisticated business applications.

One such sophisticated business application which is central to the success of any enterprise is the support of decision making. Without an effective decision support system, enterprises will be unable to exploit opportunities as they appear anywhere and anytime. For good decision making, executives and managers need to count on up-to-date, business-critical data, being instantly available on their hand-held and wireless computers. Such data are typically in the form of summarized information tailored to suit the users' analysis interests.

Traditionally, decision makers use OLAP (On-Line Analytical Processing) tools to execute decision support queries on the enterprise data warehouse or data mart. OLAP tools provide multidimensional views of data to facilitate decision making [5]. The multidimensional data model abstracts data in the form of a *data cube* where dimensions are the subject of interests (aggregated attributes) and the cell values are the measures of interest [11]. An OLAP server may store multiple summary tables (subcubes) for efficient access by queries issued by OLAP tools at the client. An interesting property of summary tables which we call *derivation dependency*, is that one summary table can be derived from one or more summary tables.

In this paper, we are addressing the issue of efficient delivery of summary tables to wireless clients (e.g., on a company wireless intranet) equipped with OLAP front-end tools. In wireless networks, broadcasting is the primary mode of operation for the physical layer. Thus, broadcasting is the natural method to propagate information in wireless links and guarantee scalability for bulk data transfer. Specifically, data can be efficiently disseminated by any combination of the following two schemes: *broadcast push* and *broadcast pull*. These exploit the asymmetry in wireless communication and the reduced energy consumption in the receiving mode. Client devices are assumed to be small and portable, and most often rely for their operation on the finite energy provided by batteries. Servers have both much larger bandwidth (downlink) available than client devices

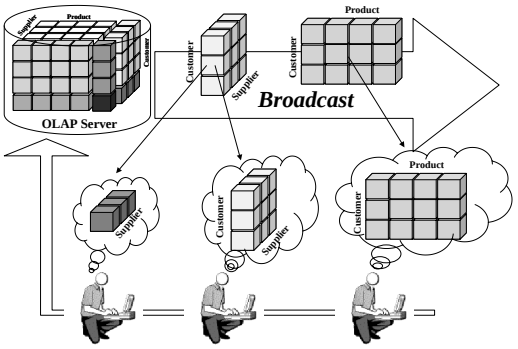

Figure 1: Wireless OLAP System

and more power to transmit large amounts of data.

In broadcast push the server repeatedly sends information to the clients without explicit client requests. Any number of clients can monitor the broadcast channel and retrieve data as they arrive on the broadcast channel. If data is properly organized to cater to the needs of the clients, such a scheme makes an effective use of the low wireless bandwidth and is ideal to achieve maximal scalability [1, 14, 12].

In broadcast pull, the clients make explicit requests for data. If multiple clients request the same data at approximately the same time, the server may aggregate these requests, and only broadcast the data once. Such a scheme also makes an effective use of the low wireless bandwidth and clearly improves user perceived performance. Several scheduling algorithms have been proposed that attempt to achieve maximum aggregation [2, 8, 21, 22].

Considering the traditional OLAP server basic functionality, the broadcast pull or *on-demand* environment as shown in Figure 1, is the most suitable for supporting wireless OLAP query processing. Interestingly, the client requests in the wireless OLAP system exhibit the above mentioned derivation dependency feature. That is, every client request is for one of the summary tables and a table requested by a client may subsume the table requested by another client. Since request aggregation is commonly used by general content delivery scheduling algorithms for efficient data dissemination, the derivation dependency property adds a new optimization dimension to the request aggregation process that allows further broadcast efficiency and scalability.

In this paper, we propose a new, heuristic, on-demand scheduling algorithm called *Subsumption-Based Scheduler* (*SBS-*α). SBS-α is non-preemptive and considers the varying sizes of the summary tables. The unique characteristic of SBS-α is its α-optimizer that exploits the derivation dependency among the summary tables to increase sharing among clients that goes beyond the exact match of requests of all the current on-demand scheduling approaches. Because each table satisfying a particular request incurs a different processing cost, SBS-α considers this cost when selecting the set of requests to be aggregated into the specific table which is broadcast at a given point. This cost is captured by the selected value of α. By considering that each different value of α yields a different scheduler, *SBS-*α can be thought of as a family of scheduling algorithms as well.

The performance of our new heuristic was evaluated experimentally using simulation. We used a non-preemptive version of the *Longest Total Stretch First* (LTSF) algorithm, both as the basis for SBS and in our comparisons. As we will argue below, from the existing approaches, LTSF promises the best performance for scheduling OLAP summary tables. Our experimental results have shown that exploiting the OLAP data cubes semantics allows SBS to achieve reductions up to 65% in average access time, while reducing average energy consumption by 16%.

The rest of this paper is organized as follows. The next section presents an overview of the related work in OLAP technologies and broadcast-based data dissemination techniques. In Section 3, we discuss our assumed wireless OLAP environment and in Section 4, we present *SBS*, our new on-demand scheduling algorithm. Our simulation testbed and experiments are presented in Sections 5 and 6, respectively.

2. BACKGROUND AND RELATED WORK

2.1 OLAP and Summary Tables

In a decision support environment, various sets of facts are analyzed along multiple dimensions. This led to the development of the multidimensional data model that represents a set of facts in a multidimensional space in a way that facilitates the generation of summarized data and reports [16]. In this model, data is typically stored using a star schema. The star schema consists of a single fact table storing the measures of interest (e.g., *sales*, or *revenue*) and a table for each dimension (e.g., *product*, *time*, or *region*).

OLAP queries typically operate on summarized, consolidated data derived from fact tables. The needed consolidated data by an OLAP query can be derived using the *data cube* operator [9]. The data cube operator is basically the union of all possible *Group-By* operators applied on the fact table. A data cube for a schema with N dimensional attributes will have 2^N possible subcubes. Given that the data cube is an expensive operator, often subcubes are precomputed and stored as summary tables at the server. Basically, a summary table can be modeled as an aggregation query, where the dimensions for analysis are the *Group-By* attributes and the measures of interest are the aggregation attributes. A detailed summary table T_d can be used to derive a more abstract one T_a. In such a case, the abstract table T_a is derivable from T_d or T_d subsumes T_a. For example, in Figure 1, by adding the measure values across *customer*, the detailed table (*supplier*, *customer*) can be used by a client to extract the abstract table (*supplier*).

The idea of using summary tables to derive one from another has been widely used in materialized views selection. The objective is to select the appropriate set of tables for materialization so that to speed up future query processing, while meeting the space constraints [11, 10]. To facilitate the selection process, the search lattice was introduced in [11]. The search lattice is a directed graph representing the subcubes space and captures the subsumption property among subcubes. For example, Figure 2 shows the lattice for the (*Supplier(S)*, *Product(P)*, *Customer(C)*) schema.

In this paper, we also use the subsumption property of summarized tables and the idea of search lattice in the scheduling of requests in order to minimize the user perceived latency.

2.2 Broadcast-Pull

Several scheduling policies have been proposed in the broadcast pull literature. These policies can be classified as either *non-preemptive* or *preemptive*. In the non-preemptive con-

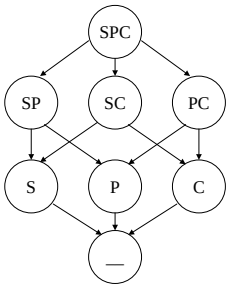

Figure 2: Data Cubes Lattice

text, it was pointed out that that *First Come First Serve* (FCFS) scheduling would provide poor access time for a broadcast pull environments [8] and *Most Requests First* (MRF) and *Longest Wait First* (LWF) were proposed as alternative efficient algorithms in [8, 22]. The *RxW* algorithm [4] combines the benefits of MRF and FCFS, where the intuition underlying RxW is that hot or popular data items are disseminated as soon as possible yet it avoids starvation of cold or less popular data items by means of an aging scheme.

Preemptive scheduling policies have been introduced to handle the heterogeneous requests problem, i.e., requests for data items of varying sizes [2]. Three preemptive algorithms have been proposed, namely, *Longest Total Stretch First* (LTSF), an offline algorithm called *BASE* and its online approximation *MAX*.

Preemptive scheduling policies exhibit better performance than the non-preemptive ones. However, preemptive schemes cannot in general support selective tuning. Selective tuning is the fundamental property for preserving energy where the main idea is, if sufficient indexing information is provided to clients, then the mobile device access pattern to the data stream can alternate between a *doze* mode waiting for data and an *active* mode tuning for required data. In a doze mode, the wireless communication equipment is powered down, and hence the mobile device is consuming power orders of magnitude less than that in the active mode. Power conservative indexing methods for single-attribute and multi-attribute based queries in broadcast push environments appeared in [14, 12, 3]. For hybrid broadcasts, the authors in [7] investigated two sets of broadcast protocols, where the requested data items are broadcast in batches and they used the techniques in [14] to index the data items on the broadcast cycle.

The idea of merging queries with overlapping answers to reduce broadcast data dissemination cost has been introduced in the context of a multicast subscription environment [6]. In this approach, a post-filtering is needed at the client side to obtain the answer to the original query. A similar proposal appeared in [17], where a semantic description is attached to broadcast unit, called a chunk, which is a cluster of data items. This allows clients to determine if a query can be answered based solely on the broadcast and to request the remaining items in the form of a supplementary query. A popularity-based scheduling policy was used to broadcast the data chunks. This assumes that the server has been informed about the clients' queries at the beginning of each broadcast cycle.

Our work carries some similarity with the work in [6, 17]. However, we are modeling an on-demand broadcast environment, where the server has no prior knowledge of the arriving requests. Additionally, in our case, the requests are for summary tables (aggregation queries) rather than queries with selection predicates as in [6, 17], where the relative sizes of dependent tables may vary tremendously.

3. WIRELESS OLAP MODEL

In this section, we are presenting our model for the wireless OLAP environment. Our assumed architecture is based on broadcast pull scheme as shown in Figure 1. The OLAP server is responsible for maintaining and disseminating the summary tables. We are assuming that all the lattice subcubes are ready at the server, which is a reasonable assumption, specially for relatively small size data marts. The Essbase system (according to [11]) is an example of commercial product that materialize all the possible summary tables.

A client sends an uplink request for a table on the *uplink channel*. Then it listens to the downlink channel for a response. A client can be in one of two modes, either a *tune* mode, listening to the broadcast, or in a *wait* mode, where the client is idle waiting for the response. Clients depend on the server to satisfy all their requests; they are not accessing any local storage and previous answers are not locally cached for future use.

An uplink request Q is characterized by the set of its Group-By attributes D. Hence, we represent a request as Q^D and the corresponding table as T^D. A summary table T^{D1} *subsumes* table T^{D2}, if $D2 \subseteq D1$, similarly, T^{D2} is *dependent* on T^{D1}. We denote the number of dimensional attributes in the set D as $|\,D\,|$ and the cardinality of table T^D as $|\,T^D\,|$.

The smallest logical unit of a broadcast is called a *packet* or *bucket*. A broadcast table is segmented into equal sized packets, where the first one is a *descriptor* packet. Every packet has a header, specifying whether it is data or descriptor packet, the offset (time step) to the beginning of the next descriptor packet, and the offset of the packet from the beginning of its descriptor packets. The descriptor packet contains a table descriptor which has an *identifier* semantically describing the table being broadcast by identifying its aggregation dimensions, the number of attribute values or tuples in the table and the number of data packets accommodating that table. We are assuming that no single data packet is occupied by tuples from different tables. Each summary table is broadcast within a broadcast cycle that starts with the table descriptor packet.

Here we used bit encoding to represent the semantics of a client request and the descriptor packet identifier. The representation is a string of bits; its length is equal to the number of the complete schema dimensions and each bit position corresponds to one of the dimensions $d_1, d_2, ..., d_n$.

If a table T^D has dimension d_x ($d_x \in D$), then the bit at position x is set to 1, otherwise it is a zero. For example, assume the (*supplier*, *product*, *customer*) schema. The representation of the (*supplier*, *customer*) summary table will be 101. This scheme can be easily extended to include tables with more than one measure and different aggregation functions. But, without loss of generality, we are assuming only one measure attribute and *sum()* as the aggregation function in this paper.

When a client submits a request for table T^R on the uplink channel, it immediately tunes to the downlink channel, and goes through a three phases access protocol: (1) *initial probe*; (2) *semantic matching*; and (3) *table retrieval*.

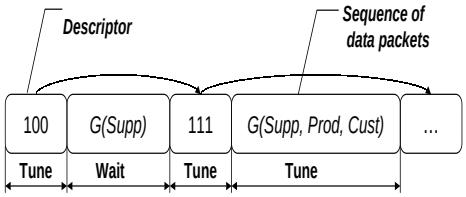

Figure 3: A Client Access to Broadcast

In the initial probe phase, the client tunes to the downlink channel and uses the nearest packet header to locate the next descriptor packet. The semantic matching phase starts when the client finds the first descriptor packet, say for table T^B, then the client can semantically classify T^B as:

1. *Exact match*: if the aggregation dimensions in T^B are the same as T^R (i.e., $R = B$).
2. *Subsumption match*: if T^B subsumes T^R, and T^B is not an exact match for T^R (i.e., $R \subset B$ and $R \neq B$).
3. *No match*: if it is neither an exact match nor a subsuming match (i.e., $R \not\subseteq B$).

For example, assume R is (*supplier*, *product*), then B_1 = (*supplier*, *product*, *customer*) is a subsumption match, while B_2 = (*product*) and B_3 = (*supplier*, *customer*) are examples of no match.

Depending on the matching result and the scheduling algorithm used (as we will see in Sections 4), the client will either switch to the final retrieval phase or it will stay in the matching one. In the former, the client stays in active mode tuning to the next sequence of data packets to read (download) table T^B. While in the latter case, it will switch to doze mode reducing power consumption. Using the offset in the packet header, it wakes up just before the next broadcast cycle (i.e., descriptor packet of the next table on broadcast) where the semantic matching process is repeated. The access protocol is shown in Figure 3.

3.1 Performance Metrics

The performance of any scheduler in a wireless environment can be expressed in terms of:

- **Access Time:** It is the user perceived latency from the time a request is posed to the time it gets the response. Its two components are the *wait time* and *tune time*.
- **Tune Time:** It is the time spent by the client listening to the downlink channel either reading a descriptor packet or a stream of data packets containing the requested summary table. During tuning, the client is in active mode.
- **Wait Time:** The total of amounts of time a client spends waiting to read a descriptor packet until it finds a matching one. A client is in doze mode during the wait time.

In active mode a client device consumes energy orders of magnitude higher than in doze mode. For this reason, tune time has been traditionally used to evaluate the power consumption of a system in a mobile environment. However, the energy dissipated in doze mode becomes more significant when the client has to wait for long intervals of time until its request is satisfied. Hence, in this paper we will adopt a weighted energy consumption cost model that includes the active and doze factors.

4. SUBSUMPTION-BASED SCHEDULER

The access profile for OLAP summary tables has the following key features:

1. Heterogeneity: summary tables are of different dimensionality (number of dimensional attributes) and varying sizes.
2. Skewed Access: Request from OLAP clients usually form a hot spot within the data cubes lattice. Most of the time queries are accessing low dimensionality tables and they often drill down for detailed ones.
3. Subsumption: it is often possible to use one detailed table to extract other summarized ones.

Hence, an appropriate scheduler should consider all of the above features with the objective of reducing access time and energy consumption. Apparently, these requirements can be satisfied using one of the preemptive algorithms proposed in [2], where the preemptive policy exhibited significant reduction in access time compared to the non-preemptive one which does not consider heterogeneity. However, as mentioned above, the use of preemption deprives a broadcast policy from deploying an effective indexing technique as in [14] which is essential for energy saving. Of the preemptive scheduling algorithms, LTSF has a corresponding non-preemptive version that retains its basic properties and can support selective tuning (in the form of basic indexing). Given our dual objective, we selected the non-preemptive LTSF as the basis of our proposed scheduler. In LTSF, the data item which has the longest total current stretch, i.e., the sum of the current stretches of all pending requests for the item, is chosen for broadcast. The current stretch of a pending request is the ratio of the time the request has been in the system thus far to its service time.

The *Subsumption-Based Scheduler* (*SBS-*α) that we are proposing in this section consists of two components: An LTSF (basic selection) component, which captures the first and second features above and the α-*optimizing* component that exploits the third feature above to control the degree of sharing.

In *SBS*, the server queues up the clients requests as they arrive. When it is time for the server to make a decision which table to broadcast next, it computes the total stretch value for each table that has at least one outstanding request. The table with the highest total stretch value (say T^{bcast}) is selected to be broadcast as in [2].

Ignoring the subsumption semantic of summary tables, a client that requested T^{req} that is derivable from T^{bcast} will wait until T^{req} is broadcast. While in an extreme case of exploring the subsumption property, the client will use T^{bcast} to derive T^{req} regardless of the relative cardinality of both tables and potentially incurring some unnecessary extra cost. Hence, we are using the parameter α to define the degree of flexibility in using the subsumption property (that captures the degree of sharing) and it works cooperatively between the server and clients as follows:

- At the server side, upon deciding the broadcast of table T^{bcast}, the server discards every pending request for a

table T^{req} that can be derived from T^{bcast} and satisfies the following property (α *rule*): The ratio of the difference in size between tables T^{bcast} and T^{req} to table T^{bcast} is less than the α value. Formally, T^{req} can be discarded and is not broadcast iff T^{bcast} is broadcast, $req \subset bcast$ and $\frac{|T^{bcast}|-|T^{req}|}{|T^{bcast}|} \leq \alpha$.

- Consequently, when a client that requested table T^{req} sees table T^{bcast} on the broadcast that subsumes T^{req} and satisfies the α rule, this client knows that its original request has been discarded by the server and it has to use T^{bcast} instead. Formally, a client requested T^{req} will use T^{bcast} iff, $req \subset bcast$ and $\frac{|T^{bcast}|-|T^{req}|}{|T^{bcast}|} \leq \alpha$.

The value of α ranges from 0 to 1. At $\alpha = 0$ there is no flexibility in using summary tables and the client access is restricted to exact match. In this case, SBS-0 is equivalent to LTSF. At $\alpha = 1$, it is the case of extreme flexibilty in which a client can use any subsuming matching table.

When implementing SBS-α, there are two alternatives of how a client can make a decision of using a subsuming match. The first one, the server encodes in the descriptor packet of T^{bcast} all the requests that can be derived from T^{bcast} and are already eliminated from the server queue. However, this scheme is not scalable and it increases the broadcast size.

The second alternative, which we adopted, lets each client make the decision locally by computing $\frac{|T^{bcast}|-|T^{req}|}{|T^{bcast}|} \leq \alpha$. The value of α is made known to the clients by including it as part of the table descriptor information along with the cardinality of T^{bcast}. A client can easily estimate the size of its own requested table T^{req} using the simple formula from [19] which only requires knowledge of the number of distinct values for each dimension. This method is particularly efficient, especially considering that the growth rate of most OLAP dimensions is very low, so these values can be downloaded once by the client and used for a long time before it needs to download it again[1].

As an example, consider the partial search lattice shown in Figure 4, in which nodes are summary tables and the number between braces is the table cardinality in units of size. Assume the search lattice nodes shown in figure, are the tables for which there exist at least one request and $\alpha = 0.9$. Also, assume that the 3-dimension table T^X (d_1, d_2, d_3) is selected for broadcast. Then, clients requests for tables (d_1, d_2) and (d_1, d_3) will be satisfied by T^X. While clients requested tables (d_1), (d_2), and (d_3) will just wait for the next broadcast cycles.

4.1 Discussion

Flexibility allows us to distinguish between the already existing on-demand broadcast scheduling algorithms that are restricted to exact match and our semantic-based *SBS-*α in the context of wireless OLAP environment. Accordingly, we are calling the algorithms mentioned in Section 2 *strict algorithms*, while the family of SBS-α, where $\alpha > 0$, are *flexible algorithms*.

The intuition for SBS is to capture all the specific features of summary tables access in an on-demand broadcast environment. The LTSF basic scheduling component encapsulates all the factors affecting access time. The α parameter controls the degree of flexibility based on the subsumption semantic property.

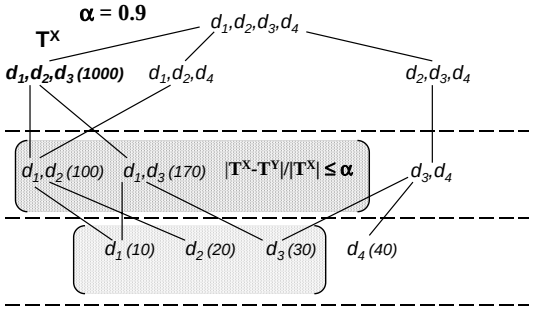

Figure 4: Flexibility

The advantage of flexibility is to find another aspect of common interest other than the exact strict one, hence decreasing the wait time and the corresponding doze energy. The drawback is the extra time a client has to spend tuning to a detailed table rather than a summarized one and the accompanying high energy consumed in the active mode. Picking a reasonable value for α will balance the trade-off between reducing the wait time (doze energy consumption) and increasing the tune time (active energy consumption). As in [11], we are assuming a linear cost model for aggregate query processing, where a table scan is required to compute the result. Hence, when extra filtering and extraction is required, it can simply overlap with the tuning phase when the client is downloading a table. As we will argue in Section 6 the energy required for processing is insignificant compared to the energy needed for wireless communication and can be ignored.

As an example for the flexibility trade-off, consider the case where T^{bcast} is selected for broadcast. The derived tables from T^{bcast} with pending requests can be classified into two groups T^{large} and T^{small} according to size. In case of request for table T^{large}, where $\frac{|T^{bcast}|-|T^{large}|}{|T^{bcast}|} \leq \alpha$. If the total stretch value for the request for table T^{large} is still not high enough, then disseminating T^{bcast} will reduce the wait time experienced by a client requested T^{large}. On the contrary, a client requested table T^{small}, where $\frac{|T^{bcast}|-|T^{small}|}{|T^{bcast}|} > \alpha$, if T^{bcast} is disseminated, the client requested T^{small} would rather wait for the next broadcast cycles to avoid the costly tune time of downloading T^{bcast}.

Let us now consider a simple numeric example that highlights the differences in scheduling decisions, average access time, and average tune time for different values of the parameter α. Table 1 shows the example settings, where there are four requests for four different tables T_1, T_2, T_3, and T_4. The A_i value represents the wait time of a request for table T_i, and S_i value is $|\ T_i\ |$. Additionally, we are assuming that tables T_2 and T_3 are derivable from T_4. The scheduler has to make the decision what is the sequence of tables to broadcast given the queue status at each broadcast cycle. In this snapshot, the four requests constitute the whole workload, i.e., no more requests will arrive at the server.

Table 2 shows the broadcast sequence (BSeq) generated by setting α to the values 0, 0.25, 0.75 (left most table is the first to be broadcast), the corresponding average access

[1]The server may also periodically broadcast the number of distinct values of each dimension along with the meta-data information about its service.

	T_1	T_2	T_3	T_4
A_i	10	4	10	29
S_i	10	25	50	60

Table 1: Example Settings

Algorithm	BSeq	AAT	ATT	BSize
SBS-0	T_1, T_4, T_2, T_3	93.25	36.25	145
SBS-0.25	T_1, T_4, T_2	74.5	38.75	95
SBS-0.75	T_1, T_4	68.25	47.5	70

Table 2: Example Results

time (AAT), average tune time (ATT) and the broadcast size (BSize). Assume that the transmission time of a table is equal to its size, for example, the transmission time for table T_4 is 60 units and its access time using SBS-0 is equal to $A_4 + S_1 + S_4$, where $(A_4 + S_1)$ is the wait time and S_4 is the tune time.

Different degrees of flexibility in using the subsumption property gave different tradeoffs between the access time and tune time. Setting α to 0, which is the basic LTSF scheduler, gave the lowest tune time but in the meantime it has the highest access time. On the contrary, setting it to 0.75 resulted in the lowest access time and the highest tune time. However, setting α to 0.25 reduced the access time by 21% and the increase in tune time is 7%.

5. EXPERIMENTAL TESTBED

We implemented a system simulation model to evaluate the potential gains using the *SBS-*α algorithm by comparing it to the non-preemptive version of LTSF, which is equivalent to SBS-0. For the clarity of presentation, the parameter α is only taking the values 0, 0.25, 0.5, 0.75, and 1. However, these values capture the extreme cases of flexibility in using the subsumption property as well as the cases of low, moderate, and high flexibility.

We modeled the environment as a single server with a set of clients. There is a single downlink broadcast channel over which all data is disseminated to the clients and a single uplink channel that clients use to send uplink requests. We are assuming that clients are able to complete any uplink request in a single uplink packet. For the purposes of this simulation, we have ignored all communication errors.

We generated a synthesized lattice for an n-dimensional data cube. The values of n is in the range between 4 and 12, with $n = 6$ be the default. The sizes of lattice subcubes is computed as in [15], where a subcube is given a binary code C. The binary code is similar to the bit encoding we used for identifying cubes on broadcast. Then the subcube size (number of tuples) is set to C^2. The final cube size is the product of the generated number of tuples and the number of attributes (dimensional and measure attributes), hence, the unit for size is the number of attribute values in a table.

We used this method to ensure diversity in subcubes sizes and significant size difference between a cube and all its dependent cubes. In the generated lattice, cubes at the bottom left area have small sizes while those at top right have larger sizes. This setting will results in 64 (2^6) possible queries for the default case.

Derived summary tables are of different sizes, i.e., they have different degrees and cardinalities. In the simulation, we are assuming that attributes values have the same sizes and a data packet capacity is 10 attribute values.

Parameter	Value
Base Cube Dimensionality	4 – 12 dimensions (default 6)
Possible Requests	16 – 4096 requests (default 64)
Packet Capacity	10 attributes values
Zipf Parameter (θ)	0.0 – 1.9 (default 0.8)
Simulation Length	100 requests/client
Number of Clients	10 – 200 clients
α-optimization	0,0.25,0.5,0.75,1

Table 3: Simulation Parameters

To test the system under a realistic workload, requests are generated by the clients according to Zipf distribution with the Zipf parameter (θ) default value is equal to 0.8. Queries are sorted according to their size, so that queries to small size tables occur with higher probability than queries to detailed ones.

We control the simulation by establishing a fixed number of requests, that is, each client was required to complete a certain number of requests before the experiment would terminate. This ensures fairness in reporting, and eliminates any possibility of reporting partially complete data. A client will pose a new request as soon as it gets an answer to its previous one. We also allow for the variability of the number of clients in the client population.

Table 3 summarizes our simulation parameters and settings. The combination of these parameters allows us to examine the scalability of the system as well as the impact of a changing workload on the algorithm performance.

6. PERFORMANCE EVALUATION

6.1 Impact of Request Rate

In this experimental setting, the number of clients varies between 1 to 200 clients, each client poses 100 requests. The variation in the number of clients reflects different request arrival rates. Requests are generated according to the previously mentioned Zipf distribution with θ equals to 0.8.

Figure 5 shows average access time for the SBS family of algorithms. For all values of α the algorithm exhibits a similar behavior, that is, the average access time increasing but ultimately stabilizing as the number of clients is increased. This behavior is the norm for broadcast data delivery to clients with shared interests. The figure shows how the access time is decreasing with increasing α for the same number of clients. Furthermore, this reduction in access time is more significant as the load increases and more flexibility is needed to handle the high request rate. For instance, consider the cases of 10 and 200 clients where $\alpha = 1$. In the case of 10 clients, the average access time decreased by 25% compared to SBS-0, while in the case of 200 clients SBS-1 achieved 65% reduction in the access time compared to the strict SBS-0.

Figure 6 shows the simulation results for our second optimization objective, i.e., the energy consumption. We express energy in terms of doze mode units assuming active:doze ratio to be 20:1 as in the ORiNOCO World PC Card [18] – the energy consumed tuning to one packet is equivalent to that consumed in dozing for 20 packets transmission time.

As expected, the extreme case of flexibility (α=1) leads to an increase in the overall energy consumptions. However, reduction in energy consumption is achieved by setting α for values less than 1. This reduction is more noticeable at higher loads where doze energy is playing an important role.

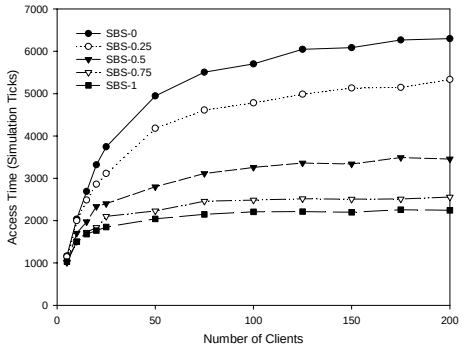

Figure 5: Average Access Time

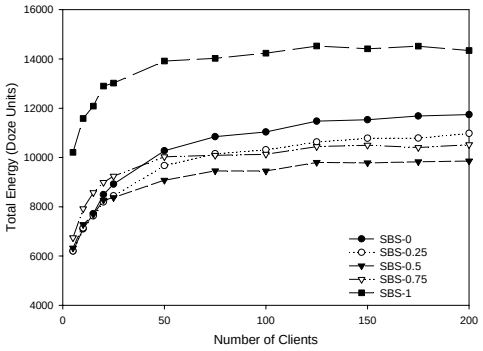

Figure 6: Average Energy Consumption

For instance, consider again the cases of 10 and 200 clients and $\alpha = 0.5$. In the case of 10 clients, energy consumption increased by 2% compared to SBS-0, while it decreased by 16% in the case of 200 clients. This gain is better illustrated in Figure 7 in which the total energy consumption is depicted by its active and doze components. At the population of 200 clients, SBS-0.5 provided a 48% reduction in doze energy compared to SBS-0, but, the active energy is increased by 17%, leading to the previously observed 16% overall reduction in energy consumption. Additionally, watching the SBS-0 performance, we can see that doze energy consumption is growing with the increase in number of clients until it becomes equally important as the active one. This explains the gains obtained at high loads, where flexibility trades a limited increase in active energy for a substantial decrease in doze energy.

6.2 Impact of Database Size

In this experiment, we are testing the influence of changing the database size on performance, specifically the number of dimensions of the OLAP data cube, which has two effects: 1) it changes the ratio between a summary table and its subsuming ones; and 2) it changes the number of generated subcubes. We experimented with number of dimensions varying from 4 to 12, while the number of clients is set to 50.

In Figure 8, for clarity of presentation, we are normalizing the flexible versions of SBS to the strict one (SBS-0). It is interesting to observe that all algorithms exhibit the same behavior, where the relative gain in access time decreases to a minimum point and then it starts increasing again. The explanation for this is that with a constant number of clients

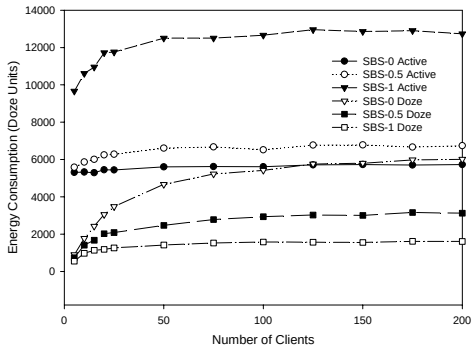

Figure 7: Active and Doze Energy Consumption

and few dimensions the chances of exact matching between queries are high and the impact of deploying flexibility is small. As the number of dimensions starts growing, the clients interests are spread in a larger lattice of subcubes. In this case, the subsuming matchings offered by flexibility adds significantly to the exact matching ones, yielding to a high degree of sharing. However, as the number of dimensions keeps growing, the frequency of subsuming matches itself is decreasing. This is the case in which attempts to aggregate requests will fail and the system should switch to serving each clients' request independently.

For instance, consider the performance of SBS-0.75 in the cases of 4 and 8 dimensions. In the 4 dimensions case, the average access time is 32% less than SBS-0, while in the 8 dimensions it provided a 55% reduction where more flexibility is used to cater the diverging clients interests. However, beyond this minimum point (8 dimensions in our experiment), the chances of the subsuming matching itself start to diminish and the performance of the flexible SBS's is getting close to that of SBS-0. Eventually at 12 dimensions, we can see the SBS family performing even worse, where taking a chance of using subsumption results in an increase in tune time that cannot be compensated by enough reduction in wait time due to lack of common interests.

To further investigate this behavior, we conducted another experiment with smaller clients population (25 Clients). The results are shown in Figure 9. As expected, by decreasing the number of clients, the minimum point moved towards the left (6 dimensions). Further, we observed a reduction in aggregation compared to the case of 50 clients which is reflected by the difference in the slopes of the curves in Figures 8 and 9. This is because with the same number of dimensions, decreasing the number of clients further reduced the degree of overlapping between interests. The two factors of reducing request rate and increasing the lattice size result in the observed difference in performance compared to Figure 8.

Figure 10 shows that when there is enough exact matching (which is similar to a light loaded system), wait time is not high enough to allow the flexibility to provide any gain in energy. But, as the dimensionality increases, using flexibility starts to significantly reduce the total energy by reducing the doze component. This behavior is sustained until a minimum point, beyond which the savings are decreasing. The explanation is that at high dimensionality, the extra active power consumption due to the flexibility is high compared to the scarce saving in doze energy.

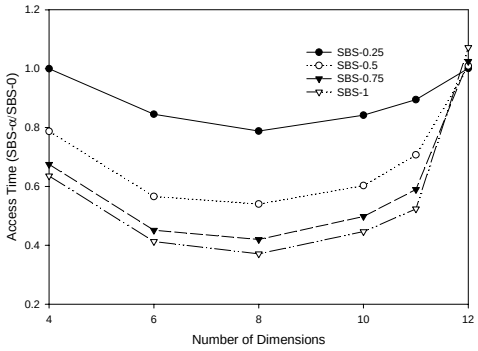

Figure 8: Latency Vs. Dimensionality (50 Clients)

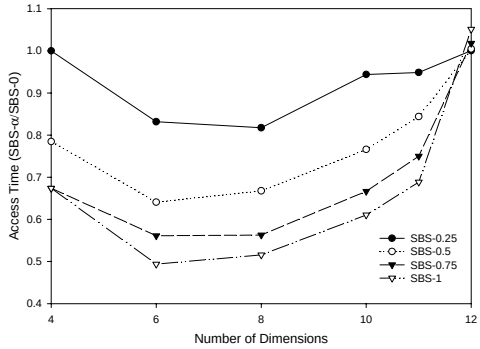

Figure 9: Latency Vs. Dimensionality (25 Clients)

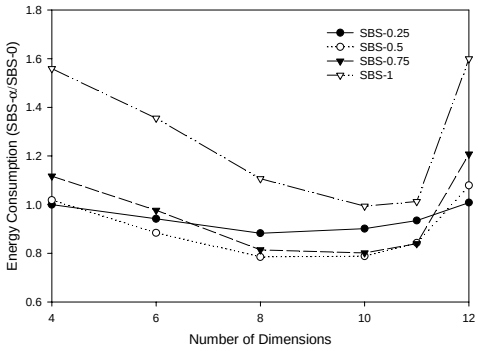

Figure 10: Energy Vs. Dimensionality

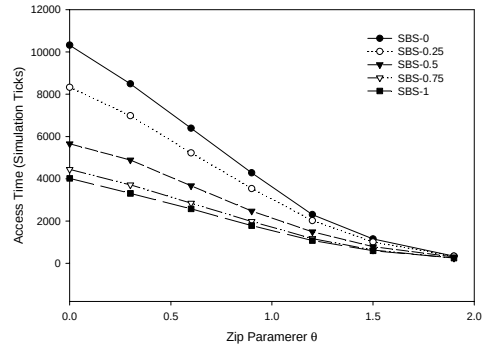

Figure 11: Access Time Vs. Zipf θ

6.3 Impact of Skewness

In all the previous comparisons, we used the default θ value of 0.8. Here, we are examining the performance for different values of θ, i.e., the degree of skewness of access. Figure 11 shows the average access time for a setting, where the number of clients equals to 50, each posing 100 requests. The Zipf parameter ranges from 0 to 1.9.

Since the number of clients (request rate) is kept constant, the increased overlap in client interests allows more efficient use of the broadcast bandwidth. Therefore, as the skew increases all algorithms provide improved reduction in access time. However, the SBS-α schedulers were $\alpha > 0$, are also taking advantage of the subsumption property between requested tables. Considering the difference in savings provided by SBS-0.75 in the cases when θ is equal to 0.9 and 1.9. In the case of $\theta = 0.9$, the average access time is reduced by 55% compared to SBS-0, however, the reduction is only 25% at $\theta = 1.9$.

The energy consumption results presented in Figure 12 came as expected. That is, the moderate use of flexibility by SBS-0.25, SBS-0.5, and SBS-0.75 showed energy reduction by tackling the doze component, whereas that reduction is diminishing at the case of highly skewed data access.

6.4 Practical Implications

In order to have a realistic insight of the savings gained by applying a semantic-based scheduling of OLAP summary tables, as well as the effect on processing cost, let us consider the practical implications in terms of time and energy units.

Consider a wireless LAN, where the broadcast channel has a bandwidth of 1Mbps. Assume each attribute value in our synthesized lattice is of size 10 bytes and each data packet capacity is 10 attribute values. It will take about 0.8 mSec to broadcast a single packet.

We assumed clients are using the IBM ThinkPad laptop [13] that is equipped with Pentium 4 mobile processor which consumes 2 Watts on average, with a 100 MHz RAM and 64 bits bus. During processing, the processor accesses the downloaded summary table from memory and the memory transfer rate determines the energy needed for processing. Hence, processing a packet will take 0.125 μSec and the processor energy consumed during this duration equals 0.125 μSec * 2 W = 0.25 μJ.

Let clients be equipped with the ORiNOCO World PC Card. The card operates on a 5V power supply, using 9mA at doze mode and 185 mA at receiver mode. Hence, dozing for one packet time will consume 0.8 mSec * 9 mA * 5 V = 36 μJ, while being active tuning to one packet will take 0.8 mSec * 185 mA * 5 V = 740 μJ.

Table 4 provides a practical numeric comparison between the the family of SBS-α algorithms. The table shows the average access time (AAT), the average communication energy consumption (ACEC), and the average processing energy consumption (APEC) per query for 100 clients.

Algorithm	AAT (Secs)	ACEC (Joules)	APEC (Joules)
SBS-0.00	4.56	0.40	0.000070
SBS-0.25	3.82	0.37	0.000072
SBS-0.50	2.6	0.34	0.000082
SBS-0.75	1.98	0.36	0.000100
SBS-1.00	1.76	0.51	0.000150

Table 4: Practical Results

As we can see from the table, SBS-0.75 is giving the best overall performance. It reduced the access time by 57%

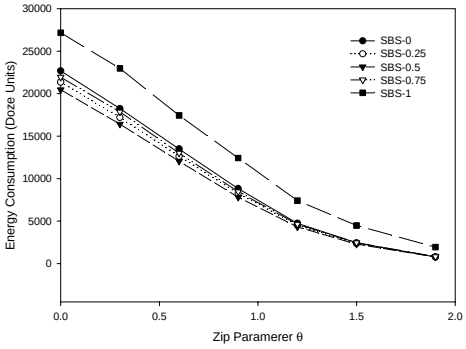

Figure 12: Energy Consumption Vs. Zipf θ

by aggregating requests for subsuming tables and letting a client derive its originally request table from a detailed one. The extra cost of active energy requirements imposed by this flexibility in scheduling tables is successfully compensated by the decrease in doze energy, even yielding a 10% reduction in total energy consumption. The table also clearly shows how insignificant the processing energy consumption is compared to the communication energy consumption. This confirms our eariler assumption of ignoring the processing energy component in calculating the overall energy consumption.

7. CONCLUSIONS

In this paper, we re-emphasized the role of broadcast based data dissemination in supporting efficient access of enterprise data warehouse and consequently enabling good decision making anytime and anywhere. Although the emphasis of our paper was on wireless and mobile computing environments, our result are applicable in wired networks which support multicasting.

More specifically, this paper has made three contributions in the context of on-demand broadcast scheduling:

- It identified the new possibility of request aggregation based on the subsumption semantics among summary tables rather than just based on the exact match of requests of all the current approaches.
- It classified on-demand scheduling algorithms into *strict* and *flexible* based on their ability to broadcast subsuming tables in respond to a given request.
- It proposed a family of heuristics called called *SBS-α*. The α-optimization parameter controls the degree of flexibility in using available subsuming tables, which provided further reductions in access time and dissipated power. The superiority of the SBS-α was demonstrated experimentally using simulation.

We are currently investigating the effect of deploying caching at the client side and evaluating the overhead at the server.

8. REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communication environments. In *Proc. of the ACM SIGMOD Conf.*, pp. 199–210, May 1995.

[2] S. Acharya and S. Muthukrishnan. Scheduling on-demand broadcasts: New metrics and algorithms. In *Proc. of Fourth Annual ACM/IEEE Conf. MobiCom*, pp. 43–54, October 1998.

[3] R. Agrawal and P. K. Chrysanthis. Efficient data dissemination to mobile clients in e-commerce applications. In *Proc. of the 3rd WECWIS*, pp. 58–65, June 2001.

[4] D. Aksoy and M. Franklin. RxW: A scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Transactions On Networking*, 7(6):846–860, December 1999.

[5] E.F. Codd. Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate. E.F. Codd and Associates, 1993.

[6] A. Crespo, O. Buyukkokten, and H. G. Molina. Efficient query subscription processing in a multicast environment (extended abstract). In *Proc. of the 16th ICDE Conf.*, February 2000.

[7] A. Datta, D. E. VanderMeer, A. Celik, and V. Kumar. Broadcast protocols to support efficient retrieval from databases by mobile users. *ACM Transactions on Database Systems (TODS)*, 24(1):1–79, 1999.

[8] H. D. Dykeman, M. Ammar, and J. W. Wong. Scheduling algorithms for videotex systems under broadcast delivery. In *Proc. of the 1986 Int'l Conf. on Communications*, pp. 1847–1851, June 1986.

[9] J. Gray, et. al. Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub Totals. In *Proc. of the ICDE Conf.*, pp. 152–159, February 1996.

[10] H. Gupta. Selection of views to materialize in a data warehouse. In *Proc. of ICDT*, pp. 98–112, Jan. 1997.

[11] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proc. of the ACM SIGMOD Conf.*, pages 205–216, June 1996.

[12] Q. Hu, W.-C. Lee, and D. L. Lee. Power conservative multi-attribute queries on data broadcast. In *Proc. of the 16th ICDE Conf.*, pp. 157–166, 2000.

[13] http://www.ibm.com

[14] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Energy efficient indexing on air. In *Proc. of the ACM SIGMOD Conf.*, pp. 25–36, May 1994.

[15] P. Kalnis, N. Mamoulis, and D. Papadias. View selection using randomized search. *DKE*, 42(1):89–111, 2002.

[16] R. Kimball. *The Data Warehouse Toolkit*. John Wiley, 1996.

[17] K. C. K. Lee, H. V. Leong, and A. Si. A semantic broadcast scheme for a mobile environment based on dynamic chunking. In *Proc. of the IEEE Int'l Conf. on Distributed Computing Systems*, pp. 522–529, 2000.

[18] ORiNOCO World PC Card. www.orinocowireless.com

[19] A. Shukla, P. M. Deshpande, J. F. Naughton, and K. Ramasamy. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *Proc. of the VLDB Conf.*, pp. 522–531, Aug. 1996.

[20] K. Stathatos, N. Roussopoulos, and J.S. Baras. Adaptive data broadcast in hybrid networks. *The VLDB Journal*, pp. 326–335, 1997.

[21] N. H. Vaidya and S. Hameed. Scheduling data broadcast in asymmetric communication environments. *ACM/Baltzer Wireless Networks*, 5(3):171–182, 1999.

[22] J. W. Wong. Broadcast delivery. *Proc. of the IEEE*, 76:1566–1577, 1988.