The Datacycle Architecture for Very High Throughput Database Systems

Gary Herman, Gita Gopal, K C Lee, and Abel Weinrib

Bell Communications Research, Inc
435 South Street, Morristown, NJ 07960-1961

ABSTRACT

The evolutionary trend toward a database-driven public communications network has motivated research into database architectures capable of executing thousands of transactions per second In this paper we introduce the Datacycle architecture, an attempt to exploit the enormous transmission bandwidth of optical systems to permit the implementation of high throughput multiprocessor database systems The architecture has the potential for unlimited query throughput, simplified data management, rapid execution of complex queries, and efficient concurrency control We describe the logical operation of the architecture and discuss implementation issues in the context of a prototype system currently under construction

1 INTRODUCTION

In this paper, we introduce the Datacycle architecture, a novel system concept which we believe can permit the implementation of very high transaction throughput database systems Our research into high throughput systems is motivated by the architectural evolution of the United States public telephone network toward a fully database-driven network In such a network, a single, logically integrated database would be consulted during call processing to establish caller permissions, perform name-address translations, execute feature logic, establish routes, etc In today's network, centralized databases are consulted only for a small fraction of calls, for example, to seek authorization to complete the call by validating a credit card number or to obtain routing instructions specific to a given call, as in an "800" call attempt In the future, we foresee a network where access to a global network database is the norm in call processing, rather than the exception

The database technology challenge posed by this view of the architecture of the public network is one of enormous transaction volume To provide 800 service, each database node in today's public network [Shei82] receives approximately 100 simple queries per second on its copy of a database of a few hundred thousand records Each record is accessed by a single key, updates are batched and executed periodically In our view of the future network architecture, processors in the network execute tens of thousands of transactions per second, some quite complex, accessing databases containing tens of millions of records, and accommodating unpredictable queries and random updates while providing consistent views to all users With the current throughput of general purpose database systems limited to below 1000 transactions per second [Gray85], database systems with the performance characteristics required by this view of network architecture do not appear achievable by conventional architectures

Our approach to achieving the required throughput seeks to exploit opportunities created by advances in processing and communications technologies Other recent proposals for new database machines have taken a similar approach, e g , through dense, inexpensive semiconductor memory in the Massive Memory Machine [Garc84] or through massive parallelism in VLSI in NON-VON [Hill86] While the Datacycle architecture also depends on VLSI technologies, the primary motivation for our approach is the abundance of inexpensive communications bandwidth which has become available through optical communications systems The Datacycle architecture is communications intensive, stimulated by the hypothesis that if costs for communications bandwidth become small, communications, processing and storage architectures for database systems should be reassessed Bandwidth abundance permits repetitive broadcast of the entire contents of the database to many processing elements, avoiding I/O bottlenecks, simplifying data administration, and achieving efficient concurrency control

2 OVERVIEW OF THE DATACYCLE ARCHITECTURE

As indicated, the strategy in the Datacycle architecture is to broadcast database contents over a very high bandwidth medium to a multitude of "listening" processors that observe the stream in parallel and use hardware filters to extract the information relevant to the transactions pending locally As illustrated conceptually in Figure 1, the entire contents of the database are read sequentially from the **storage pump**, broadcast repeatedly over the **broadcast transmission channel**, and observed passively and independently by many **record access managers**, each of which may copy records of local interest Each access manager is associated with a general purpose computing system, or **host**, on which applications reside Transactions on a host request database operations through a well-defined interface with an associated access manager The record access manager decomposes each requested operation into a set of specifications to be used to identify relevant records when they appear on the broadcast

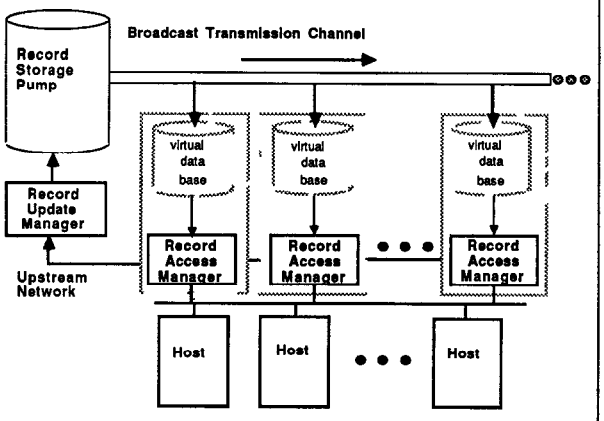

Figure 1. The Datacycle Architecture

channel, processes the records to complete the database operation, and returns the results to the host. To complete a transaction and update the database, the host submits an update/commit request to the access manager Using the **upstream network,** the access manager sends the update request to the **record update manager,** which executes a nonconflicting subset of the update requests received in a cycle The access manager subsequently monitors the database to determine if the update attempt was successful and notifies the host. In addition, the access manager can utilize the hardware monitoring capability to decentralize much of the conflict detection overhead in concurrency control For example, with "optimistic" concurrency control, the associated access manager can monitor the transaction's read set to detect potential conflicts due to transactions committing elsewhere in the system

The Datacycle architecture has similarities with information distribution architectures like teletext [Sige80] and the MIT Community Information System [Giff85], which employ broadcast of small databases or database updates to many receiving stations However, these systems are text-oriented, typically distribute relatively small amounts of data, and have relatively large access times to a given data item Further, they do not deal with the issues of updates originating at the receiving stations, transaction management, and query processing in the traditional database sense

In a database machine context, hardware filtering and broadcasting of data to alleviate the I/O bottleneck and to permit parallel access to data have been suggested before For example, in [Leil78], the data stream is searched "on the fly" as it is read from disks with parallel read heads In DIRECT [DeWi79] selected pages of the database are broadcast repetitively to multiple query processors This broadcast feature is utilized, for example, in a nested-loop join algorithm for query processing In contrast to these systems, the hardware filtering capability and the repetitive broadcast of the entire database in the Datacycle architecture are integral to both query processing and transaction management functions such as concurrency control

The Datacycle architecture has several strengths No central I/O bottleneck exists for record retrieval, regardless of the number of processing elements required to handle the transaction load. Since the entire database can be searched on the fly, *a priori* knowledge of the precise location of each record in the database should not be necessary Therefore, the overhead required for maintaining a directory can be eliminated, the system is potentially fully content-addressable Further, the content-addressability and repetitive broadcast characteristics of the architecture lend themselves to efficient execution of complex queries, such as joins These same architectural features permit efficient concurrency control mechanisms when transactions wish to modify the contents of the database Because record access managers read data and attempt writes (updates) without explicit communications among themselves, concurrency control overhead depends only on the volume of transactions and does not depend on the number of access managers (and hosts) provided to handle the transaction volume The Datacycle approach provides an opportunity to achieve high throughput, efficient execution of complex queries, and efficient concurrency control in a system with many distributed processors

An assessment of available communications, storage, and processing technologies indicates that the Datacycle architecture is technically plausible Transmission systems employing electronic modulation of semiconductor lasers at rates in excess of one gigabit per second are available commercially today Wavelength division multiplexors which can multiplex the outputs of sixteen such lasers onto a single optical fiber are also available Thus, with current technology, several gigabytes of data can be transmitted over a single optical fiber in less than a second Transmission technology is advancing rapidly, and, with the promise of future coherent optical techniques, the information capacity of a single fiber will be measured in terabits per second

Since storage access to support the broadcast function is sequential and synchronous, similar to the output of a video frame buffer, creation of the appropriate read and write bandwidths to storage should be primarily an engineering problem For example, in the case of semiconductor memories, word width, bank interleaving schemes, parallel to serial conversion, and multiplexing technologies can be chosen to achieve the desired broadcast data rate With the availability of 16 megabit memory parts, one gigabyte of memory will occupy less than a single shelf

Finally, at each listening point, the contents of the broadcast stream must be examined to identify, and copy, data items relevant to local transactions Because in the Datacycle architecture the hardware pattern comparison function is integral to both query processing and concurrency control, its efficient implementation is critical to the practical applicability of the architecture Studies of VLSI implementation of selection hardware [Faud85] are promising, projecting that VLSI technology in the 1990's will permit implementation of 200 to 500 comparators per integrated circuit, with each comparator capable of searching for a different 256 byte pattern

In the remainder of this paper, we describe the logical operation of the Datacycle architecture and discuss implementation issues in the context of an architecture prototype being built in our laboratory Specifically, Section 3 provides an overview of query handling in the Datacycle architecture, with concurrency control and recovery discussed in Sections 4 and 5 Section 6 describes our design for a prototype system. Finally, we conclude with a discussion of the domain of applicability of the architecture and describe ongoing research on the Datacycle approach to high throughput database systems

3. QUERY PROCESSING

Queries are presented to a record access manager in a high-level database language such as SQL [Date86] Within the record access manager are many hardware pattern comparators managed by a supervisory subsystem, comprising processing elements, buffers, and an interconnection network The supervisory subsystem translates queries into selection primitives that can be processed in the pattern comparator hardware The pattern comparator accepts commands of the form

Select (data type) **From** (segment) **For** (request id)
Where (expression)

The data type can be a record, or some fragment of a record, the segment defines the subset of the broadcast channel (e g , subchannel or a fragment of a subchannel) to be searched, the request id associates the instruction with a specific query, the expression defines the pattern to be matched and consists of simple boolean expressions relating the fields of a record and constants through operations such as equals, greater than, etc Using the select command to return a field or a set of fields from all records in a relation implements a project capability (without duplicate elimination) in hardware Duplicates must be detected and removed at a higher level to complete the project operation To retrieve all data items that satisfy some expression, select commands require up to one full cycle time of the database to complete

The architecture permits an efficient join algorithm To perform a join, in the first broadcast cycle of the database one pattern comparator performs a hardware project over the joining attribute on the two relations to be joined The projected values are sent to the supervisory subsystem where a data structure is built to maintain a count of the number of times each value appears in each of the two relations The data structure can be an array if the domain of the join attribute is known in advance and is relatively small Otherwise, a hash table can be used. Every value which has non-zero count for both relations will take part in the join, the count gives the size of the result exactly If enough pattern comparators are available, the supervisory subsystem loads pattern comparators with each value that will take part in the join, and in the next cycle all the desired records are retrieved. The actual join is then performed in the supervisory subsystem If fewer pattern comparators are available, more cycles may be required This join operation resembles the bit array technique in [Babb79] and the hardware-assisted semijoin approach in [Faud85] However, the hardware project capability, the cyclic nature of data presentation, and the content-addressable feature of the Datacycle architecture permit a particularly efficient implementation The algorithm has the potential to retrieve all the records required for a join in only two broadcast cycles

4. CONCURRENCY CONTROL

An efficient concurrency control mechanism is critical to achieving high throughput The concurrency control mechanism in the Datacycle architecture ensures database consistency, permits record access managers to read and write the database without explicit communications among themselves, and decentralizes much of the conflict detection for update transactions

Concurrency control is predicated on the assumptions that each broadcast cycle of the database has a distinct beginning and end and that the contents of the database within a single cycle are guaranteed to be consistent That is, the record update manager provides atomic record updates for the writes of a transaction, no partially completed updates are permitted in a cycle Thus, an access manager can execute a read-only transaction without any concurrency control overhead at all, so long as it reads all records within a single cycle

However, some scheme must be used to control the concurrent attempts of the various access managers to update the database in the storage pump Most of the techniques to control simultaneous, multiuser access to a single database fall into three categories - locking, timestamps and certification [Bern81] Each of the three techniques has a domain of applicability, i e , there are transaction environments for which each technique is best suited. In environments where the number of data items is large compared to the number of transactions running concurrently and the probability of overlapping transactions is small (network applications may fall into this category), the overhead of granting locks or maintaining timestamps does not seem justifiable In these cases certification schemes are expected to perform well The Datacycle architecture permits an efficient implementation of such a scheme, which is described here

In general, update transactions include a read phase, an execution phase, and a commit phase For the present, we consider only transactions which complete their read phase within a single cycle and, therefore, obtain a consistent read set In the read phase of a transaction, the access manager independently extracts records from the broadcast stream and maintains its own list of records in use, based on the records required locally During the execution phase of a transaction, the access manager monitors the transaction's read set For each record on the local list of records in use, the supervisory system in the access manager instructs the pattern comparator hardware to scan the broadcast stream to determine if the record has been changed (updated) since it was originally read for the associated transaction With this hardware testing capability and the repetitive broadcast of the database, the access manager can detect any modification to a pending transaction's read set, notify the host, and pass the new record values to the transaction Based on these new values, the transaction can abort, back up, or continue The decision is strictly local Unlike most certification schemes, the "certification with monitoring" scheme in the Datacycle architecture permits transactions to abort as soon as a conflict is detected, reducing processing wasted on already doomed transactions

An access manager has responsibility for detecting interference with a transaction's read set from the beginning of the transaction through the cycle in which a transaction reaches its commit phase and wishes to write to the database This cycle defines a local commit timestamp Then, the access manager sends an update request, including the new record values, the identities of records in the read set, and the commit timestamp, to the update manager

The update manager performs two types of central certification on the update requests it receives First, it validates each request against update requests granted in the recent past Some latency will exist between the time a locally certified transaction generates an update request and the time the results of the update, if granted, actually appear in the database Consequently, the update manager is responsible for detecting update attempts invalidated by previously granted updates whose effects appear in the database after the local commit timestamp, escaping detection at the access manager Second, the update manager also detects update requests that conflict with other requests submitted with the same timestamp The update manager accepts a serializable subset of attempts and introduces the changed records into pump storage Failed attempts are simply discarded by the update manager

Each access manager subsequently determines whether the transactions with submitted update requests have committed

successfully by either monitoring the actual data values or examining a portion of the bandwidth used to broadcast a log of committed transactions Commit authorization is then returned to the host

Extensions to this basic scheme for concurrency control are described in [Gopa87] For example, multiversion optimistic techniques for concurrency control can be applied in the Datacycle architecture to permit transactions to read over multiple database cycles and to provide greater overall concurrency, as well As with the simple optimistic technique described here, the multiversion approach decentralizes much of the concurrency control overhead and allows read-only transactions to complete locally

5 FAILURE RECOVERY

Graceful and rapid recovery from subsystem failure is a requirement for a practical database machine architecture The novel characteristics of the Datacycle architecture provide conceptually simple mechanisms for recovery to a known consistent state if component failure occurs The update manager and storage pump manage the entire database and must, therefore, be made very reliable Because the costs of these central subsystems are shared over the many record access managers, traditional approaches to reliability through redundancy should be economical Redundancy can lead to quick recovery, as well For example, since the database is consistent within a cycle, the last complete cycle of the database (essentially, a full database checkpoint) might be kept by a redundant "shadow" pump Thus, while the active pump is transmitting cycle K+1, the shadow pump retains the database of cycle K If the active pump fails during cycle K+1, the "shadow" pump becomes active, and the system "recovers" to the cycle K state If access managers wait to see that the full broadcast cycle is completed before authorizing transactions to commit, failure of the pump or transmission channel during cycle K+1 results in the set of pending transactions rolling back to the state existing following completion of cycle K If cycle K+1 completes successfully, the shadow copy must then be updated to reflect state K+1

Recovery from access manager failure can also be relatively simple Record access managers operate completely independently of one another; even their connection to the shared optical broadcast medium is through passive taps Hence, the failure of one access manager can have no effect on the others Also, if one of these subsystems fails, requests which would normally be served by the failed access manager can be redirected to other access managers, and the system as a whole will experience little degradation To recover from failure of an access manager with transactions in the phase between commit request and commit authorization, the update manager must maintain a log of committed transactions to be accessed when the access manager recovers or the host is rehomed to a new access manager

6 IMPLEMENTING THE DATACYCLE ARCHITECTURE

We are currently constructing a prototype of the Datacycle architecture, both to demonstrate technical feasibility and to serve as a vehicle for further research on the architecture Our efforts in the first version of the prototype focus on the record access manager, which we consider to be the critical subsystem in the architecture, both in terms of functionality and cost Our strategy for the prototype emphasizes flexibility and ease of implementation over efficiency Nevertheless, the prototype design is intended to consider the significant architectural issues to be addressed in implementing a full scale system To provide an overview of the structure and scale of the Datacycle prototype, we first briefly describe the design of the transmission channels, storage pump, and record update manager We then describe our rationale and design for the record access manager

Transmission Channels To allow for flexibility in the total size of the database, implementation of the pump and the record access manager must be independent of the total transmission capacity provided in the broadcast channel The approach, as illustrated in Figure 2, is to multiplex together a number of relatively slow data streams from the pump, carry out the transmission at the higher aggregate rate, and then, at each access manager, demultiplex the stream into a number of slower channels, each of which operates at a rate suitable for the pattern comparator technology in the access managers A need for more storage is accommodated by adding a new storage subsystem to the pump and new pattern comparator subsystems at each of the access managers Besides providing flexibility in database size, this approach also reduces the clock rate required within the pattern comparator electronics

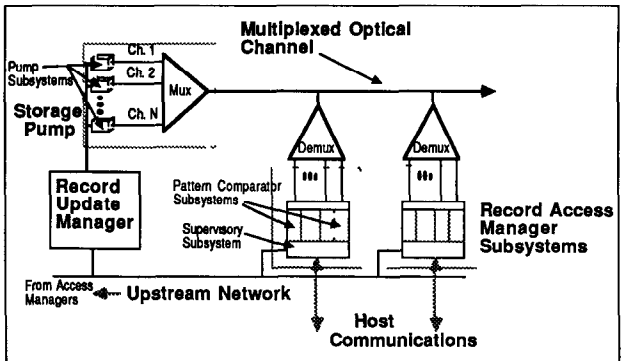

Figure 2. The Datacycle Prototype

The database prototype uses framing and transmission electronics created for an experimental broadband communications system implemented in Bellcore [Hayw87] The basic subchannel rate is 140 megabits/second, accessible for reading or writing through a 17 5 megabyte/second byte parallel interface provided by an integrated circuit framer Up to sixteen such subchannels feed cascaded 4 1 GaAs multiplexors to create an aggregate 2 24 gigabit/second bit stream that modulates a semiconductor laser The outputs of sixteen lasers can be multiplexed optically onto one single mode optical fiber Thus, the transmission technology available for the prototype could transmit a database of over four gigabytes in less than one second over a single fiber Early versions of the prototype will use 4 to 16 140 Mbps subchannels

In addition to the primary broadcast channel, an upstream channel is required to transmit record update requests from the access managers to the record update manager In the simple version of the architecture presented here, this upstream channel is separate from the broadcast medium and may be implemented by any communications network providing the required functionality and performance In our prototype, upstream communications between access managers and the update manager will be through Ethernet™

Ethernet™ is a trademark of Xerox Corporation

Storage Pump. Implementation of the record storage pump is driven by system requirements for reading records from the pump and transmitting them onto the broadcast medium, and for writing new or altered records into the pump storage to update the database An evaluation of the two most popular storage technologies today, moving head magnetic disk and semiconductor memory, indicates that semiconductor memory is a more appropriate choice for the storage pump application in terms of cost and design simplicity The sustainable read rate for most commercial drives—approximately 8 5 megabytes/sec , including seeks, platter rotation, etc , in the case of one high performance drive [Fuji84]—is too low relative to storage volume That is, such a drive can transmit less than 3% of its capacity in a second, in the Datacycle architecture, the extra capacity is wasted On the other hand, generating the appropriate I/O bandwidths for synchronous, sequential read access to semiconductor memory is a relatively straightforward design exercise The major disadvantage of semiconductor memory as primary storage for a database application is its volatility, which affects strategies for system reliability and recovery (c f , [Hagm86])

In the Datacycle prototype, the storage pump will consist of a number of 4 megabyte banks of semiconductor memory, with each bank managed by a microprocessor, as shown in Figure 3 Using a DMA-like address sequencer, each bank will, in turn, create a 17 5 megabytes/second data stream to feed the framer interface Dual-ported memory in each bank provides the mechanism for the update manager to provide new records to pump storage and to instruct the bank microprocessor to delete old records The bank microprocessor is responsible for maintaining the memory bank as instructed by the update manager Assuming that four daisy-chained banks feed a single framer, each 140 Mbps transmission subchannel broadcasts 16 megabytes repeated every 0 9 seconds At any time in a broadcast cycle, only one fourth of the banks will be in "broadcast" mode, in the other banks, the bank microprocessors will be adding and deleting records, performing storage housekeeping duties, etc

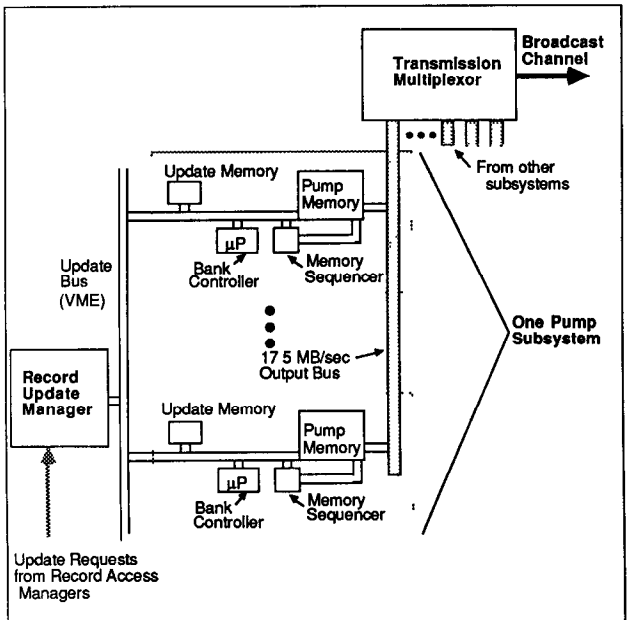

Figure 3 Prototype Pump and Update Manager

Record Update Manager. The update manager in the Datacycle prototype is a general purpose microprocessor which receives update requests from the access managers over a local area network and provides altered records to the cell microprocessors through shared memory The initial algorithm for resolving update conflicts is simply first-come, first-served For each cycle, the update manager maintains tables for the read sets and write sets of granted updates Record IDs of incoming update requests are checked against the tables, if a non-serializable conflict is detected, the update is ignored. If the atomicity of the update to storage can be guaranteed, it is executed, and the tables are updated to include the new read and write set If the update atomicity cannot be guaranteed, due to lack of time remaining in the cycle, the update is ignored. Update throughput can be improved, at the cost of delaying transaction completion, by allowing one or more additional broadcast cycles between update request and confirmation Thus, an update request submitted during cycle K might be introduced in the database during cycle K+n. The update manager has responsibility for detecting conflicts between each new update attempt and updates which were granted previously but which were not visible to the access manager

The critical design issue for the pump and update manager is how to provide the guarantee of atomic updates for each transaction without constraining the update throughput to too low a level This is an area of current research that is clearly critical to the successful realization of the architecture

Record Access Manager. The access manager provides its host or hosts with the functionality of a typical database machine, decomposing requests into database primitives, assigning the primitives to pattern comparator hardware, and then processing the returned data items to generate the response, as described in Section 3 The access manager simultaneously supervises the concurrency control process, one view of which was described in Section 4

The access manager is the focus for our initial research because of its novel use of both the pattern comparator hardware and the repetitive broadcast of the database contents for database queries and for concurrency control The architecture is critically dependent on the efficiency of the pattern comparison operation For example, a selection may require dedicating pattern comparators for an exhaustive search of every subchannel of the database Monitoring for concurrency control also increases the demand for pattern comparators, as do complex queries such as joins

If pattern comparator hardware with the required boolean functions is capable of hundreds of simultaneous comparisons per IC, as [Faud85] projects, then simple, exhaustive search strategies are plausible On the other hand, if pattern comparators are expensive, then more sophisticated and hardware-efficient query processing, concurrency control, and content searching strategies may be required. For example, the available pattern comparators might be more efficiently used by utilizing the channel structure together with some indexing or hashing scheme so that only some of the channels need to be searched for a single pattern Further, segments defined by sections of time on a channel can also be used in the indexing

Our prototype record access manager is designed to provide us an environment in which to experiment with alternative approaches in implementing key functions in the Datacycle architecture In the Datacycle prototype, the access manager consists of distinct subsystems, the Query Manager (QM), Data Manager (DM), and Intelligent Record Retrieval Modules

(IRs), as shown in Figure 4 In the model of Section 3, the QM and DM comprise the supervisory subsystem, and the IRs contain the pattern comparators

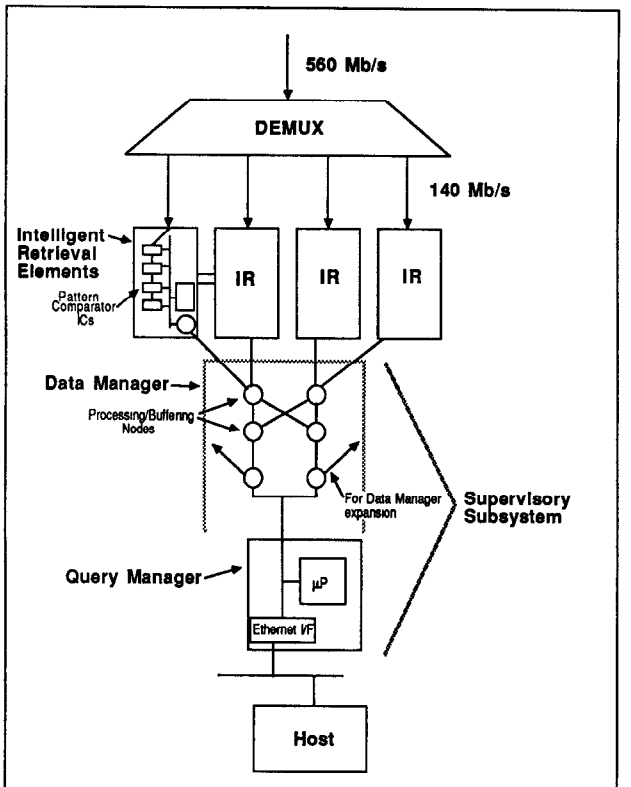

Figure 4 Record Access Manager Prototype

The Query Manager receives queries from a host, decomposes those queries into operating primitives, and submits the primitives to the DM for execution The Query Manager also manages the updates for committing transactions, initiating the update request to the update manager and, in conjunction with the DMs and IRs, confirming the success of the update In our prototype implementation, the QM is a UNIX™ -based microcomputer which communicates with the host and the update manager using Ethernet

As shown in Figure 4, the Data Manager in the prototype consists of multiple processing nodes connected to form a network expandable to manage access to many broadcast subchannels The DM maps the logical data space (based on relation names and field values) to the physical data space (based on subchannels and subchannel segments), routes the primitives to the appropriate IRs, buffers intermediate results, and routes the results of the primitive database operations to the Query Manager

Finally, each IR consists of a processor managing four VLSI pattern comparator units that perform the low level search operations in the architecture The pattern comparator ICs are being designed for initial implementation in 2 0 micron CMOS Their anticipated capabilities include variable length predicates, capacity to store up to 150 (four byte) predicates on chip, a throughput of 20 million simple predicate tests per second, capability to evaluate simple boolean expressions, and the ability to switch between target patterns on database segment boundaries Pattern comparator ICs receive data from the transmission subsystem over the 17 5 megabyte per second byte parallel interfaces provided by framer ICs

7. DISCUSSION

As a system concept, the Datacycle architecture has attractive characteristics, both for our original network application and as a general purpose database system Still, the architecture has limitations and weaknesses, and both technical feasibility and practicality remain unproven In this section, we discuss our current view of the domain of applicability of the architecture and identify key open issues and areas of current research

The architecture appears well suited for very high throughput, read-oriented applications in which transactions have relatively small, predeclared read sets and wish to access a common database through any of several keys The "database-driven network" problem which motivated our research has these characteristics The maximum size of a database suitable for the Datacycle approach is a subject of current investigation, for a fixed broadcast cycle time, a larger database requires a higher bandwidth channel and, consequently, either more pattern comparators (to search more subchannels) or faster comparators While databases of hundreds of megabytes appear realizable with sub-second access times using current transmission technology, the dominant cost of a large database system using the Datacycle architecture is the hardware cost of searching

While the Datacycle approach is "read-oriented", it is not a "read-only" architecture like teletext. Use of the repetitive broadcast of the database to decentralize conflict detection and synchronize the actions of the many access managers is a key feature of architecture The architecture should exhibit excellent update throughput when compared with conventional architectures While the centralized update manager remains an ultimate throughput bottleneck in the architecture for transactions which write, as well as read, the database, we are investigating architectures which distribute the update certification function to achieve even greater throughput, applying the approach of [Wein87]

An issue in the Datacycle architecture is support for transactions which have large read sets and/or extended execution phases, such transactions may have difficulty completing successfully To address this issue, we are exploring concurrency control techniques which increase the proportion of transactions which complete successfully in a high conflict environment [Gopa87] One approach is to employ a multiversion scheme, permitting consistent reads to occur over multiple cycles and increasing concurrency, at the cost of increased storage and bandwidth requirements Other approaches may exploit the hardware monitoring capability at the access managers to permit finer granularity for local validation, predicate-based validation, and recovery from conflict by partial transaction backup

A weakness of the architecture is the access latency which results from the synchronous, periodic presentation of data This read latency adversely affects the system response time for transactions that consist of a series of dependent reads, that is, where each database read depends on the result of its predecessor A transaction with K such reads applied to a database with cycle time T can take KT or longer to complete Two approaches may reduce this problem first, structure the database and query processing language to anticipate such queries, and, second, drive down the cycle time T As before,

UNIX™ is a trademark of AT&T Bell Laboratories

reducing T requires a higher bandwidth channel and more investment in search hardware, i e , either more, or faster, pattern comparators

The cost of the hardware pattern comparison operation is critical to query processing, database search strategies, and concurrency control in the Datacycle architecture Through the VLSI design activity in our prototype, we are investigating alternative approaches to achieving highly parallel selection operations in VLSI, technologies to permit very high speed comparisons are also desirable, since they permit shortening the database cycle time and reducing the number of subchannels required for very large databases We are also exploring techniques exploiting *a priori* knowledge of the data placement in subchannels and subchannel segments to utilize hardware pattern comparators more efficiently, while retaining most of the simplicity of data management promised by the content-addressable feature of the architecture

Research on query decomposition explores the dependency of decomposition strategy and access manager architecture on the functionality and cost of the hardware selection operations More functionality (e g , wild card capability, boolean expressions) can reduce the pattern comparator's speed or capacity for simultaneous searches On the other hand, less functionality results in less efficient filtering of the data stream, increasing the I/O and processing burden on the supervisory subsystem

8 SUMMARY

In this paper, we have described a novel architecture for high throughput database systems based on exploitation of abundant communications bandwidth In this architecture, repetitive broadcast of the entire contents of a database to many processing elements eliminates read bottlenecks, simplifies data management, supports rapid execution of complex queries, and permits efficient concurrency control The architecture is plausible given current transmission, storage, and VLSI technologies, a system prototype is under construction

9 ACKNOWLEDGEMENTS

Many of the concepts described in this paper draw on the contributions of past and present members of the Datacycle project Eric Gausmann, Kalwant Grewal, Dorothy DeLuca, Craig Reding, and David Wagner Also, we acknowledge with gratitude the comments and suggestions of our colleagues elsewhere in Bellcore

10 REFERENCES

[Babb79] Babb, E "Implementing a Relational Database by Means of Specialized Hardware," *ACM Transactions on Database Systems*, Vol 4, No 1, March 1979,1-29

[Bern81] Bernstein, P A and Goodman, N "Concurrency Control in Distributed Database Systems," *ACM Computing Surveys*, Vol 12, No 2, June 1981, 185-221

[Date86] Date, C J *An Introduction to Database Systems Volume 1, 4th Edition* Reading Addision-Wesley Publishing Co , 1986

[DeWi79] DeWitt, D J "DIRECT - A Multiprocessor Organization for Supporting Relational Database Management Systems," *IEEE Transactions on Computers*, Vol c-28, No 6, June 1979, 395-406

[Faud85] Faudemay, P , and Valduriez, P "Design and Analysis of a Direct Filter Using Parallel Comparators," *The 4th International Workshop on Database Machines*, D J DeWitt and H Boral (ed), Grand Bahama Island, March 1985

[Fuji84] "M2350A OEM Parallel Data Transfer Disk Drive," Fujitsu America, Inc , May 1984

[Garc84] Garcia-Molina, H , Lipton, R J , and Valdes, J "A Massive Memory Machine," *IEEE Transactions on Computers*, Vol C-23, No 5, May 1984

[Giff85] Gifford, D K , Baldwin, R W , Berlin, S T , and Lucassen, J M "An Architecture for Large Scale Information Systems," *ACM Operating Systems Review*, Vol 19, No 5, December 1985

[Gopa87] Gopal, G , Herman, G , and Weinrib, A "Concurrency Control in the Datacycle Architecture " Bell Communications Research Technical Memorandum, TM-ARH-008936 Submitted to the 13th Conference on Very Large Databases, September 1987

[Gray85] Gray, J , Good, B , Gawlick, D , Homer, P , and Sammer, H "One Thousand Transactions Per Second", *Proceedings of IEEE COMPCON 1985*

[Hagm86] Hagman, R B "A Crash Recovery Scheme for a Memory-Resident Database System," *IEEE Transactions on Computers*, Vol C-35, No 9, September 1986

[Hayw87] Hayward, G , Linnell, L , Mahoney, D , and Smoot, L "A Broadband ISDN Local Access System Using Emerging Technology Components," *Proceedings of the 12th International Switching Symposium*, April 1987, Phoenix, Arizona

[Hill86] Hillyer, B K , and Shaw, D E "NON-VON's Performance on Certain Database Benchmarks," *IEEE Transactions on Software Engineering*, Vol 12, No 4, April 1986, 577-582

[Leil78] Leilich, H -O , Stiege, G , and Zeidler, H Ch "A Search Processor for Data Base Management Systems," *Proceedings of the 4th Conference on Very Large Databases*, June 1978, 280-287

[Shei82] Sheinbein, D , and Weber, R P "800 Service Using SPC Network Capability," *Bell System Technical Journal*, Vol 61, No 7, September 1982, 1737-1744

[Sige80] Sigel, Efrem *Videotext The Coming Revolution in Home/Office Information Retrieveal*, White Plains Knowledge Industry Publications, 1980

[Wein87] Weinrib, A , and Gopal, G "Decentralized Resource Allocation for Distributed Systems," *Proceedings of INFOCOM 87*, April 1987, San Francisco