PANEL

THE EFFECT OF LARGE MAIN MEMORY ON DATABASE SYSTEMS

Dina Bitton

Department of Computer Science
Cornell University, Ithaca NY 14853

Panelists.

Hector Garcia-Molina,
Princeton University
Dieter Gawlick, Amdahl
David Lomet, Wang Institute

1. The Issue

The availability of inexpensive, large main memories coupled with the demand for faster response time are bringing a new perspective to database technology Designers of database systems are reconsidering the assumption that databases must reside on disk during transaction processing Subatantial performance gains can be achieved by making a large portion of, or the entire database, reside in main memory One approach is to use very large buffers to improve conventional disk access methods A more radical approach is to view the database as part of the user's address space in main memory, rather than as a collection of mass-storage files This leads to the advent of Main Memory Database Systems (**MMDBS**'s), which exploit memory residency of the database in their schemes for physical data organization, query optimization, concurrency control and crash recovery

This panel will address the merits and problems associated with using very large I/O buffers and mapping the database to main memory Here are some important questions that one may ask about the effect of large memories on database technology

(1) Will the database of the future be entirely mapped to main memory ?

(2) If so, should a MMDBS rely on the host operating system's virtual memory management?

(3) If not, to what extent are our current techniques for file management and access methods applicable in the presence of very large buffers?

(4) What types of data structures and access methods are best for MMDBS's?

(5) How do we manage transactions and recover from crashes when the primary copy of the database resides in memory?

(6) How do we evaluate and measure the performance of a MMDBS?

We will briefly put these problems in perspective, and outline the panelists' viewpoints

2. Large Buffer Pools

In the design of conventional disk database management systems (DDBS's), file organization, access methods and buffer management are designed to reduce the number of disk accesses Performance is enhanced by providing a large buffer pool where data records may be prefetched, or frequently accessed data, such as indices or hash tables, can be kept The acquisition of large memories for this buffer pool can be evaluated in terms of performance gain per dollar-cost In a recent study, Gray proposes the "Five-Minute Rule" for determining a cost-effective size for a main memory buffer pool Every database page that is referenced every five minutes should be memory resident [Gr85]

3. Main Memory Databases

There are applications that cannot tolerate delays caused by access to disk storage (at 30 msec or more per block access) For instance, meeting tight bounds on response time in real time systems or

certain transaction systems may require eliminating I/O time Another environment where main memory databases may be viable is an office workstation, where (a) memory is relatively cheap, (b) the user expects a database system to respond as quickly as an interactive editor [AHK85] Clearly, when the ratio of main memory to disk capacity is higher (typically 1 10 in workstations, in contrast to at least 1 100 in mainframes) and disks are slower, the advantage of storing the database on disk diminishes Thus, there exist applications for which the cost of acquiring enough memory for a MMDBS can be justified

4. Access Methods: Space-Time Optimization

A number of recent studies investigate the implication of the memory residency assumption on the design of database systems [KM84, DKO84, AHK85, Sh85, LC85] In particular, analytical models and simulation are used in [DKO84, Sh85, LC85] to evaluate access methods and join algorithms under this assumption Clearly, access methods and query processing algorithms that are efficient for DDBS's may not have the same advantages in a MMDBS In main memory, space efficiency is critical Thus a prevalent access method such as the B-Tree may not be advantageous for a main memory database, because it stores twice as many key values and pointers as other index methods Likewise, sequential file access, as supported by B^+-Trees, loses its attractiveness when the data resides in memory Finally, fast query processing algorithms that create large intermediate results, such as a sort-merge join, are not appropriate when memory residency must be preserved

In a MMDBS, the query optimizer must optimize both processing time and memory utilization Sometimes, it may have to trade time for space For instance, the database system code must be kept compact in order to leave space for operand relations and intermediate computations Under these conditions, it may be preferable to deal with less special cases in query optimization and restrict the choice to a few well tested access plans

Another critical parameter in query optimization is the allocation of memory for the result of an operation Exact estimates of the size of a temporary relation (e g a join) are critical, since large errors may result in substantial performance degradation or even failure of the MMDBS, due to limitations in the size of the address space or memory fragmentation

5. Transaction Management

Concurrency control and crash recovery are hard problems in MMDBS's Although our theoretical understanding of serializability and commitment remains applicable in the context of main memory databases, transaction management requires new algorithms when the primary copy of the database resides in memory Research in this area is on-going at Princeton University, and at IBM Yorktown

6. Performance Evaluation

In benchmarking MMDBS's, many of the techniques previously used in benchmarking DDBS's [BDT83] are inadequate because certain performance parameters are specifically related to the main memory residence assumption In particular, space requirements, in the form of virtual and real memory, and memory management, by the operating system and the database system, are critical parameters in the design of a MMDBS benchmark Appropriate performance metrics for MMDBS's should be based on the **Space-Time Integral** of a set of representative test queries To compute this integral, memory use can be measured precisely using memory reference traces, or reliable lower and upper bounds for the integral can be computed using the virtual memory requirements of queries [BT86b]

Building of the first MMDBS prototypes will undoubtedly contribute to a better understanding of performance tradeoffs The database system that supports the integrated office system **OBE** (**O**ffice-**B**y-**E**xample) is one such example [Zl82, WAB86] A benchmark of OBE, reported in [BT86a], identifies some of the issues that must be considered in the design and implementation of MMDBS's

7. Panelists' Viewpoints

Crash Recovery in a MMDBS
(Hector Garcia-Molina)

If the primary copy of the database resides permanently in main memory, then crash recovery is a critical problem One must examine various options for managing a backup copy on disk without paying a high price in performance Options that are being investigated in the framework of the Massive Memory Machine project at Princeton University, range from conventional logging, to group commits, to dedicated logging hardware and non-volatile areas of memory

Memory Management
(Dieter Gawlick)

The history of main memory databases goes back to IMS/VS Fast Path (1976) A modern MMDBS should

(1) use addressing that is transparent to the location of data This has to be supported by the hardware and the operating system

(2) be aware of storage needs at different storage levels cache, byte-addressable memory, page-addressable memory, external devices

(3) have an interface to the operating system that allows the database system to give hints on the usage of data to the operating system, and allows the operating system to optimize the overall performance

Access Methods
(David Lomet)

The availability of very large memories results in significant performance gains for access methods in database systems Access method performance for a random probe can be reduced from the current 2+ disk I/O's currently required for search, to one or perhaps fewer Most commercial databases **will not** be entirely memory resident Hence, main memory will be exploited more opportunistically through the use of buffering or virtual memory File organization should continue to be block oriented, with perhaps increased concern about storage utilization Transaction recovery under these conditions will have very important performance consequences

REFERENCES

[AHK85] Ammann, A , Hanrahan, M , and Krishnamurthy, R , "Design of a Memory Resident DBMS", *Proceedings of IEEE COMPCON 1985*

[BDT83] Bitton, D , DeWitt, D and Turbyfill, C , "Benchmarking Database Systems - A Systematic Approach", *Proceedings of VLDB 1983*

[BT86a] Bitton, D , and Turbyfill, C , "Performance Evaluation of Main Memory Database Systems," *Cornell University TR 86-731*

[BT86b] Bitton, D , and Turbyfill, C , "Space-Time Performance Metrics for Main Memory Database Systems, A Case Study," *Submitted to VLDB 1986*

[DKO84] DeWitt, D , Katz, R , Olken, F , Shapiro, L , Stonebraker, M , and Wood, D , "Implementation Techniques for Main Memory Database Systems", *Proceedings of SIGMOD 1984*

[Gr85] Gray, J , "The 5 Minute Rule," *Technical Note*, Tandem Computers, May 1985

[LC85] Lehman, T J , and Carey, M J , "A Study of Index Structures for Main Memory Database Systems," *University of Wisconsin TR 605*, July 1985

[Sh85] Shapiro, L D , "Join Processing in Database Systems with Large Memories," *North Dakota State University TR*, December 1985

[Wh85] Whang, K Y , "Query Optimization in Office-By-Example," *IBM RC 11571*, December 1985

[WAB86] Whang, K Y , Ammann, A , Bolmarcich, T , et al, "Office-By-Example, An Integrated Office System and Database Manager," *IBM RC*, 1986

[Zl82] Zloof,M , "Office-By-Example A Business Language that Unifies Data and Word Processing and Electronic Mail," *IBM Sys. Journal*, 21 3, 1982