International Journal of Computer and Communication Technology

Volume 4 | Issue 2 Article 12

April 2013

Review on Main Memory Database

P. A. Deshmukh
Deptt. of Computer Technology, Nagpur University, PCE, Nagpur, preeti.deshmukh8@gmail.com

Available online at www.interscience.in

Review on Main Memory Database

P. A. Deshmukh
Deptt. of Computer Technology, Nagpur University, PCE, Nagpur.
E-mail : preeti.deshmukh8@gmail.com

Abstract **-** The idea of using Main Memory Database (MMDB) as physical memory is not new but is in existence quite since a decade. MMDB have evolved from a period when they were only used for caching or in high-speed data systems to a time now in twenty first century when they form a established part of the mainstream IT. Early in this century, although larger main memories were affordable but processors were not fast enough for main memory databases to be admired. However, today's processors are faster, available in multicore and multiprocessor configurations having 64-bit memory addressability stocked with multiple gigabytes of main memory. Thus, MMDBs definitely call for a solution for meeting the requirements of next generation IT challenges. To aid this swing, database systems are reconsidered to handle implementation issues adjoining the inherent differences between disk and memory storage and gain performance benefits. This paper is a review on Main Memory Databases
(MMDB).

Keywords **-** *Main memory, MMDB, DRDB, SolidDB, TimesTen*

I. INTRODUCTION

Most real-time applications need very short and anticipated response time and Main Memory as we know has short response time. The decreasing cost of Main Memory consequently makes it affordable and suitable for such applications.[1] MMDB eliminates disk access by storing and manipulating entire database in main memory. It is also known as in-memory database system (IMDBS), main memory database (MMDB) or real-time database (RTDB). For performance-significant systems MMDB offer very low response time and very high throughput.[1] MMDB do away with the overhead of handling multiple disk locations and managing memory buffers thus reducing CPU work. MMDB altogether changed the basic fundamental postulation leading to research and design of each and every component of the traditional disk based management system. Thus MMDB brought upon important implication in Data Representation, Data Access Algorithms, Query Processing, Recovery, Concurrency control and the like. The major attracting benefits of MMDB's are accelerated transactions, high reliability, data integrity, Multi-User Concurrency with consistent response times. Major limitation of MMDB is its volatile nature and therefore issues of database recovery are more complex than in traditional DBMS systems [4]. By introducing special purpose hardware such as battery-backed up memory boards, uninterruptable power supplies, error detecting and correcting memory, and triple modular redundancy, reliability is improved. Though, this only reduces the probability of media failure. So, just as for DRDB, backups are required possibly to tape or other disks likewise MMDB always have to have a backup copy of the database, probably on disk [2]. MMDB indeed has emerged distinctively to meet the needs of embedded systems. As a matter of fact MMDB have flourish in recent times and have advance from an era when they were only used for caching, or in high-speed data systems, to a time now in 2011 when they may form a far more widespread part of the IT. Telecom and networking are the two major industries where specialized versions of MMDB technology are widely used. To name a few MMDB products from ancient pencil-and paper designs (MM-DBMS, MARS, HALO) to prototype or tested implementations (OBE, TPK, System M) to commercial systems (Fast Path) to most recent available in market are IBM's DB2 UDB Server, Netcool Object Server and Object Grid MMDB, ASE 1 5.5 I n - Memory Database and Oracle TimesTen IMDB. The main asset of a MMDB is its unparalleled speed for querying and update. It turns out that simple data structures like the binary AVL tree, T-Tree, and simple bucket-chained hash outperform disk-based structures like B-tree and linear hash, due to the fact that the only costs involved in index lookup and maintenance are CPU and memory access. The T-Tree is an order-preserving tree structure designed specifically for use in main memory whose primary goal is to reduce overall computation time while using as little memory as possible [3]. But query optimization in MMDB is major issue of concern. One challenge in this area is to model the interaction between coding style, hardware factors like CPU and memory architecture and query parameters into a reliable prediction of main memory execution cost[1]. This paper simply brings upon a review on all this issues of MMDB by highlighting its history, its need, its applications, its advantages and disadvantages and most importantly future scope. To explain perception of MMDB we converse on Oracle’s TimesTen and IBM’s SolidDB.

II. EVOLUTION OF MMDB

Indeed MMDB have evolved from a period when they were only used for caching or in high-speed data systems to a time now in twenty first century when they form a established part of the mainstream IT. In recent times MMDB has become known distinctively for real time systems and embedded systems. It was during the mid 80s as the outlay of main memories was dropping, the idea of keeping large database occupant in main memory flourished specifically for high-speed data systems. However the end of popularity of MMDB techniques came in the early 1990s, when it became clear that not only DRAM sizes had grown, but also disk size, and data sizes. With the advent of real-time applications seeking very short and anticipated response time, MMDB thereafter were only considered for real-time database applications like embedded systems or telephone switches and critical applications like financial services, defense industries and Communications. In the present day, however not only main memory sizes in commodity computers continue to increase but also processors has evolved to multiple processors and multiple cores per processor in many systems thus growing altogether in orders of magnitude faster than they were just a decade ago. These trade off compel to predict a dazzling potential for MMDB. And it will not be juvenile to expect that the traditional way of on-disk data base might vanish just like the mails replaced by e-mails. Over the years several database management systems for memory resident data have been proposed and implemented. To name a few prehistoric one from MM-DBMS, MARS, HALO a pencil-and-paper designs to archetype implementations OBE, TPK, System M to commercial systems Fast Path and recently Oracle’s TimesTen, IBM’s SolidDB, Sybase ASE(Adaptive Server Enterprise) and Open source MMDB also exist such as Fast DB, Monet DB, H2 and HSQLDB. SAP is coming up with a column-based, in-memory database to get faster and less expensive answers to database mining queries.

III. CALL FOR MMDB

If we look at the trade off at some point early in this century, although larger main memories were affordable but processors were not fast enough for main memory databases to be admired. However, today’s processors are faster, available in multicore and multiprocessor configurations having 64-bit memory addressability stocked with multiple gigabytes of main memory. Also accessibility of internal data management DBMS software that better exploits memory adds to attraction of large MMDB execution. [5] Thus, we can say that MMDB usage can now be practical to deliver better performance and respond flexibly to amplified demand at low cost. MMDBs definitely call for a solution for meeting the requirements of next generation IT challenges.

IV. MMDB ARCHITECTURE

The Oracle TimesTen In-Memory Database is preferred for performance-critical systems. It runs in the application tier, close to applications, and optionally in process with applications. It can be used as the standalone database or as a cache to the Oracle Database[7]

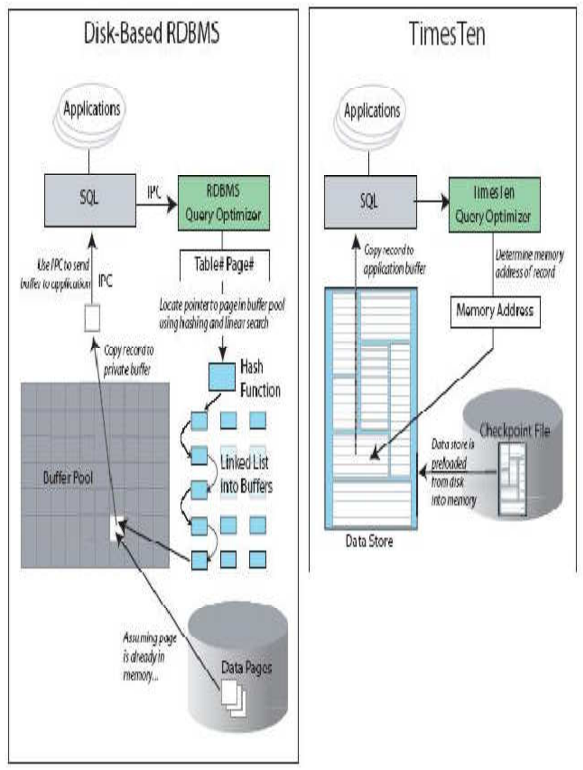

Disk-Based RDBMS
Applications
SQL
IPC
RDBMS Query Optimizer
Table# Page#
Use IPC to send buffer to application
IPC
Locate pointer to page in buffer pool using hashing and linear search
Copy record to private buffer
Hash Function
Buffer Pool
Linked List into Buffers
Assuming page is already in memory...
Data Pages
TimesTen
Applications
SQL
TimesTen Query Optimizer
Copy record to application buffer
Determine memory address of record
Memory Address
Checkpoint File
Data store is preloaded from disk into memory
Data Store

FIG 1: COMPARING DRDB AND ORACLE TIMES TEN MMDB[8]

TimesTen is designed for MMDB, takes more direct routes to data, reduces the length of the code path and implifies algorithms and structure eventually reducing the complexity. Buffer pool management totally disappears, number of machine instructions are reduced the structure and size of index pages is simplified, consequently the design becomes simple and more compact and most importantly requests are executed faster. Figure 1 shows the simplicity of the TimesTen design.[8]

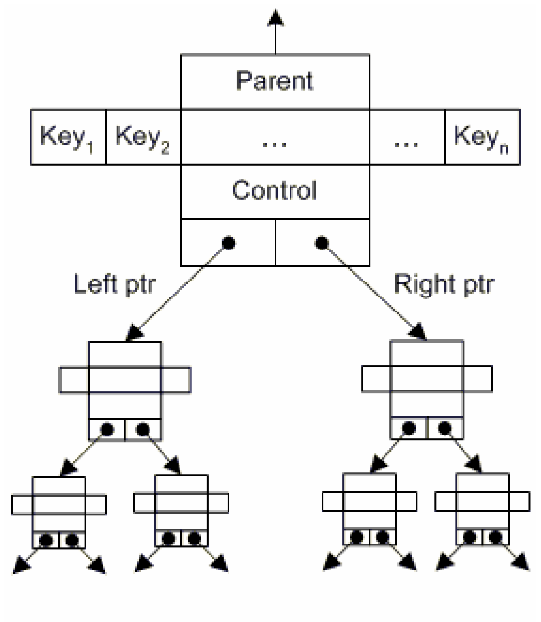
V. WORKING OF MMDB

1. DATA REPRESENTATION

Relational data are usually represented as flat files. Tuples are stored sequentially. Enumerated types larger than the pointer size are stored in the tuple as pointers to the domain table values, domain tables can be shared among different columns and even among different relations.

2. INDICES USED

MMDB uses T-tree index structure unlike B-tree index structure used by DRDB. Since the ultimate aim of MMDB is to condense computation time while exploiting little memory. T-tree index structure is explicitly designed for MMDB. A T-tree node consists of ordered elements in the range min and max values, and two pointers to the left and right nodes



d) T-tree

T-trees uphold the fact that the actual data is always in main memory collectively with the index, hence it do not keep copies of actual attribute values within the index tree nodes. Instead it just contains pointers to the actual data fields [1]. It is an ordered structure like an AVL tree having multiple keys per node. It is an Ideal index structure for ordered search over data. Other index structure supported by MMDB is heap file for handling a large number of fixed-length data items. Hash file supports unordered scan of data items as well as locking of data item that are obtained transparently when items are inserted, deleted, updated or scanned. The Oracle TimesTen, uses T-tree and hash indexing algorithms to speed access to indexed data, while also reducing CPU consumption. Use of T-trees dramatically reduces the CPU processing required to access data and completely eliminates the index value compression and expansion found in B-trees.

3. APPLICATION PROGRAMMING INTERFACE & PERFORMANCE

MMDB supports SQL industry standards for data processing and applications uses JDBC (Java database connectivity) or ODBC (open database connectivity) interface for issuing SQL (structured query language) commands. The database definition, replication, configurations also exercise SQL syntax rule. In MMDB, the transaction requesting an object is directly given its actual memory position by totally removing the concept of private buffers used in DRDB. It significantly improves the performance. The performance of a MMDB database manager depends primarily on processing time unlike DRDB where performance is determined by the count of I/O operations.[1] Nevertheless, at present, the buffering subsystems of DRDB has become effective enough to minimize the disk I/O but at the cost of complexity and excessive CPU utilization. However, MMDB (Eg:-TimesTen) performs the same database operations using 1/10th of the CPU instructions, resulting in a tremendous improvement in both throughput and response time. TimesTen uses simple and fast direct pointers to records in memory. TimesTen's in-memory data management offers several features viz compact memory structures, streamlined data retrieval, memory optimized indexing and precise optimization.

4. ACCOMPLISHING ACID PROPERTY

MMDB supports transactions that comply to ACID (atomic, consistent, isolated and durable) property to access the data. MMDB supports ANSI serializablity allowing greater concurrency. In TimesTen, as each transaction progresses, it records its data modifications in an in-memory log. At commit time, the relevant portion of the log is flushed to disk. This log flush operation makes that transaction and all previously-committed transactions durable. TimesTen allows applications to choose the transaction features they need so they do not incur the performance overhead of features they do not need. TimesTen achieves ACID property by providing features like guaranteed atomicity and durability, guaranteed atomicity and delayed durability, guaranteed atomicity but no guaranteed durability, no guaranteed atomicity and no guaranteed durability, controlling durability and logging using durable commits. [8]

5. QUERY OPTIMIZATION

Since disk access is not a factor in MMDB, the optimization cost model includes factors such as the cost of evaluating predicates, presence of indexes, the cardinality of tables and the presence of ORDER BY clauses in the query thus choosing the best query plan. Optimizer cost sensitivity is somewhat higher in MMDB than in disk-based systems. TimesTen and IMDB Cache provide range, hash and bitmap indexes and support two types of join methods nested-loop and merge-join. The optimizer can create temporary indexes as needed. The optimizer also accepts hints that give applications the flexibility to make tradeoffs between such factors as temporary space usage and performance.

6. CONCURRENCY CONTROL

Since access to main memory is much faster than disk access, transactions complete more quickly. Lock contention are not as important as it is for DRDB. To achieve maximum concurrency TimesTen facilitate rowlevellocking. Although it also permits a transaction to obtain a lock on an entire table if doing so may improve performance for intended applications. Row-level locking is worthy for most applications, as it endows the finest granularity of concurrency control. SolidDB offers two different concurrency control mechanisms, pessimistic (always conflict) and optimistic(never conflict).Main-memory tables (M-tables) are always pessimistic. The optimistic mode is about not waiting for the locks at all. That increases concurrency but requires more programming. The pessimistic mode with the READ COMMITTED isolation level provides as much concurrency as required by the intended application.

7. RECOVERY

MMDB are logically more exposed to failure than DRDB since it has to assure the high-performance requirement of many real-time applications. Therefore recovery system is a major concern for MMDB. Logging, Checkpoint and Reloading are the measures that ensures recovery of MMDB from any failure. Durability of database is achieved by logging changes from committed transactions to secondary storage and making frequent updates to a disk image of the database. TimesTen replicates the entire TimesTen database to one or more TimesTen nodes. On failure where the standby node becomes the active node, the failed node can be recovered from the standby (now active). IBM's SolidDB realize transactional durability by keeping two separate but synchronized copies of the database at all times as well as storing log files on-disk. In the event of a failure, the recovery happens to the standby database in less than one second without data loss.

VI. APPLICATIONS OF MMDB

The MMDB are particularly useful for the development of the embedded operating systems, embedded software, for testing and developing applications, for processing transient data etc. MMDSs running on RTOSs(real-time operating systems) is the best option for applications like IP network routing, telecom switching. It is also very useful in set-top boxes, managing MP3 player music databases. MMDB is also widely being accepted for non embedded applications viz trading, social networking sites, e-commerce and the list is going to increase at a rapid rate in near future.

VII. SOME OF MMDB AVAILABLE IN MARKET

1. Oracle's TimesTen is a in-memory relational database software. TimesTen is designed for low latency, high-volume data and transaction management.
2. IBM's solidDB ver 7 for the Power7 architecture applications have access to 8 terabytes of memory per core, and up to 2 petabytes of memory globally, ObjectGrid - a grid-enabled, in-memory database for applications that are written in Java and Tivoli Netcool ObjectServer, which is the Netcool inmemory database for Tivoli Netcool/Webtop to collect and process high volumes of management data.
3. Alcatel-Lucent: DataBlitz
4. Birdstep Technology formerly Raima Corporation's RDM Embedded - Raima Database Manager , RDM Server, RDM Mobile.
5. Enea's Polyhedra - a family of fault-tolerant, inmemory, transactional RDBMSs, available in 32-bit and 64-bit versions for a variety of platforms.
6. McObject LLC: eXtremeDB Embedded Database CSQL (Open Source) supported by sourceforge.net, FastDB (Open Source): it does not support a client server architecture, MonetDB (Open Source)
7. HSQLDB is a Open Source has a memory-only mode supported by sourceforge.net.

VIII. PROBLEMS IN FRONT OF MMDB

Universally, main memory in a computer normally around 4 GB to 8 GB while a server with a large amount of RAM today would be in the range of 32 to 64GB range, whereas large databases requirement is in terabyte range, this divergence makes MMDB unrealistic for databases of a larger size. It therefore seems that instead of replacing existing DRDB with MMDB, better to complement DRDB with MMDB by caching a specific set of tables. If a failure occurs on the server running in MMDB, to ensure recoverability, MMDB will log transactions to a disk. MMDB offer to allow performance to be balanced at the cost of data loss by reducing the frequency of logging. Yet again this suits only certain types of applications while for many database any form of transaction loss is unacceptable. As TimesTen's data will dwell in memory, the size of the database will be limited to the amount of RAM available on the dedicated computer. However certain applications do have TimesTen databases as large as one terabyte. Nevertheless, the number of users is certainly higher than ever before. Since 80s to until recently, CPU speed improved at an annual rate of fifty to fifty-five percent while memory speed only improved about ten percent. It is expected that memory latency would become devastating bottleneck in performance of computer. The growing disproportion of speed between CPU and memory is current research topic. Standard main-memory database technology has mainly ignored this hardware development. The design of algorithms and especially cost models is still based on the assumptions that did hold in the early 80's.

IX. CONCLUSIONS

The dominance of database management on disk system is on the verge to disappear. In the coming years MMDB's effectiveness and the affordability as well as abundance of the processors and memory will compel the database community to make this shift over. MMDB is not only faster than disk-based database system but also eliminates the need for the use of tier 1 storage in database deployment significantly reducing operational costs associated with the database storage. While making this shift over, the users should consider the data they intend to move to MMDB as well as the frequent operations performed on the database. It is definitely not hyperbolic to conclude that main memory will be seen as the prime storage of a database and disks will serve only as recovery system.

REFERENCES

[1] Fahimeh Raja, Niloofar Razavi, Melody Siadaty, "A comparison study of Main memory database and Disk Resident System"

[2] Hector Gracia-Molina, Kenneth Salem, "Main Memory Database System:An Overview" IEEE transaction on Knowledge and Data Engineering, Vol 4 No.6, Dec 1992

[3] Stefen Manegold, "Understanding ,Modelling, and Improving Main-Memory Database Performance" , Nov. 2002.

[4] Margaret H. Eich, "Main Memory Database Recovery", IEEE 1986

[5] Carl W. Olofson, "Breaking the Disk Barrier with In-Memory DBMS Technology: Sybase Adds a Big Performance Boost for ASE 15.5, January 2010

[6] Piyush Burte, Boanerges Aleman-Meza, D. Brent Weatherly, Rong Wu, "Transaction Management for a main memory database".

[7] Djellel E. Difallah Luke Thayer, "Main memory database system(ppt)", 3/6/2010

[8] Oracle TimesTen In-Memory Database Operations Guide Release 6.0

[9] IBM solidDB IBM solidDB Universal Cache Version 6.5 In- Memory Database User Guide.

[10] Abolfazl Aleahmad, Hadi Am, Masoud Rahgoza, "Main Memory Databases vs. Disk-Resident Databases"

[11] David J.Dewitt, Randy H. Katz, Frank Olken, Leonard D.Shapiro, Michael R.Stonebraker, David Wood, "Implementation Techniques for Main Memory Database System", CSTR#529, January 1984.

[12] George Copeland, Tom Keller, Ravi Krishnamurthy and Marc Smith MCC, "The Case For Safe RAM".

[13] Rajeev Rastogi S. Seshadri, Philip Bohannon, Dennis Leinbaughl, Avi Silberschatz, S. Sudarshan, "Logical and Physical Versioning in Main Memory Databases" Proceedings of the 23rd VLDB Conference Athens, Greece, 1997

[14] McObject LLC, "In-Memory Database Systems: Myths and Facts", Copyright 2009, McObject LLC

[15] Tobin J. Lehman, Michael J. Carey, "A Study of Index Structures for Main Memory Database Management Systems", Proceedings of the Twelfth International Conference on Very Large Data Bases, Kyoto, August, 1986

❑❑❑