University of Melbourne

Advanced Database Systems

COMP90050

Main Memory Database Management Systems

Authors:
Hammad Javed
Ivn Patricio Valarezo
Saman Bonab

ID: 601099

May 12, 2013

Abstract

Main Memory Databases (MMDB), which are also referred to as In Memory Databases (IMDB), store their complete data inside RAM which adds to performance of the applications linked to the database. These special type of databases are often misinterpreted as the conventional on disk database, which is optimized for disk I/O, residing in the physical memory. However in contrast, the MMDBs use other techniques for performance in data structures, access methods and durability. These types of databases are best suited for real time embedded applications as well as Business Enterprise solutions which process massive amounts of data. This report discusses briefly the limitations of traditional on disk or Disk Resident Databases (DRDB), optimization techniques used by MMDBs to overcome these limitations including data organization along with methods to ensure reliability within MMDBs.

1 Introduction

Since the inception of computing there are mainly two trends which has significant effects on IT (King, 2011).According to Moores law (Gray & Reuter, 1993) the processing power doubles every 18 to 24 months, this trend has held for almost half a century with no sign of fading.

As the increase in the processing power came with decrease in cost, the number of people adapting to computing has soared with leaps. This necessitates the evolution in the field and resulted in the creation of a whole set of smart and intelligent devices to perform the computations.

More computations and involvements in almost every strata of life from medical to military and space means more data to be processed and its exponential increase. Which tends to be the other driving trend in IT. This implies the processing of larger data by applications and in some cases where speed and time is crucial (real time systems).

Unfortunately the ever increase in processing speed is not linear with respect to other computing components especially storage systems (Boncz, Manegold, & Kersten, 1999). In fact the storage is the bottleneck for their high latency for high volume data processing real time applications.

Hard Disk drives are the cheaper, scalable and reliable solution for data storage but due to mechanical parts involved in the data access it has considerable latency issues as compared to other storage media and thus lies at the bottom of the storage hierarchy in terms of performance (Figure 1).

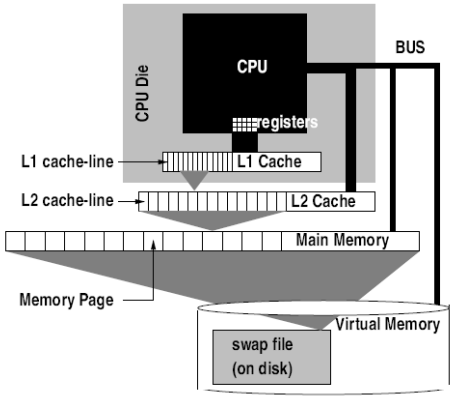

Figure 1: Hierarchical Memory System

RAM or DRAM lies below high speed L1/L2 cache or SRAM but above any other storage medium in memory hierarchy system categorized as increase in performance from bottom to top as shown in figure 1. Since the SRAM cannot be extended in size due to hardware limitations therefore DRAM or RAM is the ideal candidate for data storage.

Solid State Drives (SSDs) are nowadays another popular and widely used data storage devices in systems all over the world. Embedded and real time systems and even personal computers and laptops utilize their low latency data access features for specific applications. Since no moving part is included in SSDs, they outperform the hard disks in terms of disk I/O operations resulting in better responsiveness (McObject, 2009). Even though these NAND based devices are faster and better in performance than hard disks but they are nowhere near to DRAMs amazing speeds (Research, 2010).

Definition As the name implies Main Memory Databases MMDB or In Memory Databases (IMDB) is a database management system which stores the complete database on physical memory or RAM.

1.1 History

Most people argue in memory computing to be a relatively new and unproven technology. This is a myth rather than fact. For ages caching mechanisms is being used in traditional DRDBs which involves the most frequent use data inside RAM for fast access (McObject, 2009) . In addition in-memory tables

feature have also been part of some DBMS to hold data inside Main Memory. Also the use of DRDBs in main memory was a practice carried out to get better response times.

This entails that MMDB are not new and considerable work has been done in this regard in 1980s. However they were not popular as the cost of memory and other limitations related to persistence of data were considered a problem. Many experimental MMDBs were developed then like MARS, HALO and MM-DBMS as part of research along with commercial systems like IBM Fastpath (Garcia-Molina & Salem, 1992).

As of today IBM, Oracle and SAP being the top database vendors in the market have acquired the companies which were developing and marketing in memory database solutions (Metz, 2011). Gartner[1] has placed In Memory computing as part of *Top ten technology trends for 2013* (Penchikala, 2013). This clearly indicates the increase in the demand of MMDBs in market and their adaptation to the solutions in coming years.

1.2 Bounds of MMDBs

MMDBs with their high performance mechanisms are the perfect contenders for embedded systems (Graves, 2002). These systems with the evolution of smarter devices and expanding feature set needs to manage complex data structures which while written from ground up will require extra effort. With MMDBs pointer style low latency data access and simpler optimization techniques the embedded system developers have less things to worry about, stepping MMDBs adaptation to these types of systems. These systems include set-top boxes, network switches and consumer electronics.

The real-time systems require time-critical access to and processing of massive amounts data. Example applications that handle large amounts of data and have stringent timing requirements include telephone switching, radar tracking and others. Furthermore as stated by (Kao & Garcia-Molina, 1993), the real time database system design must not contain latency components like disk I/O operations, massage passing or garbage collection. Thus MMDBs architecture fits perfectly into the real-time application scenario.

The role of the MMDBs has greatly expanded over the last couple of years as high-end 64-bit servers can now accommodate databases that are quite large even terabytes in size. Therefore the applications for large IMDBs further include many popular large-scale Web applications and social networking sites.

Nowadays, the data within an enterprise is distributed throughout a wide range of applications and stored in disjoint locations. Analytical reporting with amalgamated view on the data as a whole is a cumbersome and time-consuming process. Besides these reports are not entirely based on the operational data, but on combined data from a data warehouse by transforming it through ETL[2]process. Hasso has pointed out in (Plattner & Zeier, 2011) that In-Memory data management assists the analytical operations resulting in planning, forecasting and faster decision making by separating it from operational data processing. Thus resulting in innovative and hybrid style of enterprise applications residing partially or fully in memory.

2 MMDB vs. DRDB : A Comparison

A Main Memory Database Management System eliminates the most expensive Disk I/O operations which are the bottle neck in traditional DRDBs. The complete database resides inside RAM. However the question here arises *Putting a DRDB in main memory will deliver us the same performance results as a MMDB* or *MMDBs are the same traditional databases inside RAM*. This notion is not true at all. Although the traditional DRDB inside main memory out performs its instance on disk but while its comparison with pure MMDB the performance gap is far too much (McObject, 2012). Linux systems and now windows as well have the capability to create a RAM disk (File system inside the main memory). But a traditional DRDB deployed on such a virtually fast hard drive doesn't provide the same benefits of a pure MMDB. In-memory databases are less complex than their on disk counterparts fully deployed in physical memory and thus require minor usage of processing cycles and RAM.

Another popular feature which most database providers provide in their DBMSs are the creation of memory tables, through which certain tables can be designated for all-in-memory handling (McObject,

[1]Gartner, Inc. is an American information technology research and advisory firm head-quartered in Stamford, Connecticut, United States

[2]ETL (Extract, Transform and Load) is a process in data warehousing responsible for pulling data out of the source systems and placing it into a data warehouse

2009). So does this mean that *with these memory features, DRDBs can compete with MMDBs performance?*. Not really as the problem remains the same as memory tables not only don't change the database design assumptions but also have certain additional restrictions. For example, in MySQL, memory tables cannot contain BLOB or TEXT columns and the maximum table size is fixed. Furthermore in contrast to MMDBs, the space that is freed up by deletion of tuples in an in-memory relation can only be used by the same relation, which is wastage of precious space inside RAM.

Comparing MMDBs with DRDBs we can find at least 4 key differences:

- Cache Maintenance
- Data Transfer
- Transaction Logging
- Concurrency Control

2.1 Cache Maintenance

2.2 Data Transfer

2.3 Transaction Logging

2.4 Concurrency Control

3 Characteristics/Strong points of MMDB

4 Issues in MMDBs

5 Background and Analysis

5.1 Concepts

We need to put concepts, I've found a good source from (Kemper and Neumann (2011))

5.2 Technology awareness

Let's put here something related with the main concerns that leads to this emerging technology. Also, its important to be aware that the ACID properties of DB Systems should be contrasted. In terms of each one of its meanings.

Atomicity In order to achieve atomicity, the IMDB should be able to handle the effects of unsuccessful transactions. In general terms, the problem has to explicitly with the data existent in the volatile memory, since all successful transaction are in a successful state only after been committed to the logging infrastructure.

Durability It is clear that during a failure, the IMDB Systems has the most obvious disadvantage, (without mention the hardware based solutions like batteries and so on), so the effects of a commit must be restored on a failure, this is the principle of Durability of course. One of the ideas to accomplish this is use *redo Logging* .

6 Data Organization

Relational Databases have been the backbone of business applications for more than 20 years, trying to provide companies with a management information system for a set of applications. During all this time, we have dreamed with the possibility of having all the information at our fingertip, we even have sold this idea(Plattner, 2009).

But due to many reasons, we have not been able to offer this. And our systems have been separated in two main different groups: The online transaction processing (OLTP) and on line analytical processing (OLAP).

6.1 OLTP and OLAP

Under the OLTP classification, could be grouped systems with a high rate of day to day transactions, most of the database systems (as we know them) are mainly used for transaction processing. Some examples of OLTP systems are financial, sales, Manufacturing, order entry, banking transaction processing, human resources. All this systems performs relatively well mainly because they work on a small portion of the data. The TCP-C benchmark publish an average processing of 100.000 such sales transactions per second on a powerful system (Kemper & Neumann, 2011).

On the other hand, analytical, business intelligence and financial planning application were moved out to separated systems (for more flexibility and better performance). The OLAP Systems could be aimed toward specific task related to the company data warehousing.

Even though, this solutions are different in context, both are still based on *Relational theory* but with different technical approaches. The main purposes of this separation could be generalized into *performance* and maybe *technical* issues. Even though both systems keep the essence in terms of relational theory, there are some important differences between them (Plattner, 2009).

Moreover, recent development in the field of OLAP and the increased availability of main memory (sufficient enough to hold a completed compressed database) have enabled the processing of complex analytical requests in a fraction of a second and thus ease the development of new business processes and applications. The next step seems obviously to undo the separation between OLTP and OLAP and all requests be handled on a combined data set.

Now that the main memory is abundant, we have gone back to see the possibility of having a all-in-one MMDB. SAP[3] is one of the most enthusiastic companies pushing the use of Main Memory Databases for a mixed OLTP & OLAP environment (Plattner, 2009), the company advertises SAP HANA as a generic MMDB solutions.

In this direction, it is suitable to estimate the challenges toward this join in technical perspectives, specifically, we have defined two main critical points related to data organization: The improved MMDB indexing system and the data Storage.

6.2 MMDB Indexing

It is mandatory to talk first about the differences between MMDB and on-disk DB, at first instance: In a MMDB Index, the main goal is to redesign the data structure and algorithm to make efficient use of *CPU* and *memory space* rather than minimize disk access, since we don't have disk of course. On the other hand, disk-oriented index structure are designed to minimize disk access and minimize disk space, this mainly because traditional database systems are CPU bounded because they spend considerable effort to avoid disk accesses.

A Main Memory oriented index structure is designed to reduce overall computation while using as little memory as possible (Lehman & Carey, 1986), since everything is in memory, MMDB Index could store only pointers to tuples or structures in the main memory (the actual data) instead of storing attribute values as on-disk DB usually does, decreasing efficiently the index footprint.

Although a MMDB system could be arranged in different ways, two main types of *Main Memory Index Structure* could be specified: Order Preserving index structures and randomized (Lehman & Carey, 1986). Under the order preserving group could be summarized: arrays, AVL Trees, B Trees and T-Trees. T-Tree is one of the most important and well know structure, named T-Tree after its 'T' shape. On the other randomized group falls: Chained Bucket Hashing, Linear Hashing and Extendible Hashing.

Although, the performance observed in a MMDB could be outstanding compared to a on disk DB, the index structure is a critical bottleneck (Leis, Kemper, & Neumann, n.d.). The T-Tree (and one of the most developed) index technology involved in MMDB was proposed 25 years ago, and was designed based on the AVL and B Trees. Acording to Leis et al. (n.d.), "the dramatic processor architecture changes have rendered T-trees, like all traditional binary search trees, inefficient on modern hardware".

TODO: maybe talk about details of t-trees...

6.2.1 Column-Store Database System

Column-oriented databases are based on vertical partitioning by columns, this is not a new idea, Column-oriented store where considered around 70s. The core of the functioning is based in the fact that each column is stored separately, having the logical scheme built around unique positioning keys (Krueger

[3]SAP AG: A German multinational software corporation that makes enterprise software to manage business operations and customer relations.

et al., 2011). This vertical data organization offers particular advantages to reading fewer attributes. According to (Plattner, 2009), in real world applications, "only 10% of the attributes of a single table are typically used in one SQL statement", this could imply that: if during requests no unnecessary columns must be read, we could just use the columns independently of the related column attributes. Row oriented structures on hard drives normally can't perform this independence.

As a first premise, recent improvements in parallel processing of modern CPUs, could lead to think that a column oriented approach today seems to be ideal. The Column oriented storage has the following advantages:

- Column Store performs outstanding on modern CPUS.
- The Storage model is based on vertical fragmentation.
- Column Store Allow data compression
- Better memory consumption performance
- Data arranged in columns are better suited for parallel processing because of its independence.

Although, column oriented storage presents a lot of interesting features, it's worth to mention that there are also some drawbacks that kept this technology behind row oriented storage: for example: Column stores are expensive to update, but also having all data in main memory overcomes this limitation. Also column store increases seek time, but this is a concern only for on-disk DB systems.

From (Plattner, 2009), it seems that although more memory has been always useful, the database systems for OLTP where not well adopted in parallel environments, one of the reasons of this were problems like deadlocks and temporary locking in parallel transactions. So in general terms has not been a good idea. Really? review this since Plattner is one of the pro Column Oriented store.

Also from (Plattner, 2009), initial tests regarding in-memory databases using relational type based on *row storage* has not shown notable performance over RDBMSs. The opportunity for *column based storage* now raises due the abundant availability of main memory.

Compression is another characteristic that Column-oriented ease. The redundancy of column store and the homogeneous domain is an advantage and a convenience for compression techniques. All the data within a column belongs to the same type, and the entropy[4] is relatively low.

TODO: talk more about column oriented, put some graphics

6.3 IMDB Data Recovery

7 Conclusions

7.1 Conclusion regarding OLTP and OLAP merging

We could talk about: *MonetDB* , *VoltDB*

References

Boncz, P., Manegold, S., & Kersten, M. (1999). Database architecture optimized for the new bottleneck: Memory access. In *Proceedings of the international conference on very large data bases* (pp. 54–65).

Garcia-Molina, H., & Salem, K. (1992). Main memory database systems: An overview. *Knowledge and Data Engineering, IEEE Transactions on*, *4*(6), 509–516.

Graves, S. (2002). In-memory database systems. *Linux Journal*, *2002*(101), 10.

Gray, J., & Reuter, A. (1993). *Transaction processing*. Address: Kaufmann.

Kao, B., & Garcia-Molina, H. (1993). *An overview of real-time database systems* (Technical Report No. 1993-6). Stanford University. Available from `http://ilpubs.stanford.edu:8090/39/`

Kemper, A., & Neumann, T. (2011). Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots. *2011 IEEE 27th International Conference on Data Engineering (ICDE)*, 195. Available from `https://ezp.lib.unimelb.edu.au/login?url=https://search-ebscohost-com.ezp.lib.unimelb.edu.au/login.aspx?direct=true\&db=edb\&AN=80280785\&site=eds-live`

[4]The entropy could be defined as the similarities of data

King, E. (2011). *The growth and expanding application of in-memory databases.* www.loyola.edu/department/lattanze/working-papers.aspx.

Krueger, J., Huebner, F., Wust, J., Boissier, M., Zeier, A., & Plattner, H. (2011). Main memory databases for enterprise applications. In *Industrial engineering and engineering management (ie&em), 2011 ieee 18th international conference on* (pp. 547–557). Potsdam Germany: University of Potsdam.

Lehman, T. J., & Carey, M. J. (1986). A study of index structures for main memory database management systems. In *Conference on very large data bases* (Vol. 294). University of Wisconsin, Madison, WI 53706: Computer Science Department.

Leis, V., Kemper, A., & Neumann, T. (n.d.). The adaptive radix tree: Artful indexing for main-memory databases. *Technische Universitat Munchen*.

McObject. (2009). *In-memory database systems:myths and facts.*

McObject. (2012). *In-memory vs. ram-disk database systems:a linux-based benchmark.*

Metz, C. (2011). *Say hello to memory. its the new disk.* http://www.wired.com/wiredenterprise/2011/12/memory-is-the-new-disk.

Penchikala, S. (2013). *Gartner's technology trends for information infrastructure: Big data, nosql and in-memory computing.* http://www.infoq.com/news/2013/04/gartner-technology-trends.

Plattner, H. (2009). A common database approach for oltp and olap using an in-memory column database. In *Proceedings of the 2009 acm sigmod international conference on management of data* (pp. 1–2). New York, NY, USA: ACM. Available from `http://doi.acm.org/10.1145/1559845.1559846`

Plattner, H., & Zeier, A. (2011). *In-memory data management: an inflection point for enterprise applications.* Springerverlag Berlin Heidelberg.

Research, I. (2010). *The state of solid state storage* (Tech. Rep.).