University of Melbourne

Advanced Database Systems

COMP90050

Main Memory Database Management Systems

Authors:
Hammad
Ivn Patricio Valarezo
Saman

ID: 601099

May 10, 2013

Abstract

Here we need to put the abstract of the work done.

1 Introduction

The main Introduction.

2 Background and Analysis

2.1 Concepts

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

2.2 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* .

3 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).

3.1 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[1] 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.

3.2 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

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

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...

3.2.1 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 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[2] is relatively low.

TODO: talk more about column oriented, put some graphics

3.3 IMDB Data Recovery

4 Conclusions

4.1 Conclusion regarding OLTP and OLAP merging

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

References

Abadi, D. J., Boncz, P. A., & Harizopoulos, S. (2009, August). Column-oriented database systems. *Proc. VLDB Endow.*, *2*(2), 1664–1665. Available from `http://dl.acm.org.ezp.lib.unimelb.edu.au/citation.cfm?id=1687553.1687625`

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`

Krishnamurthi, M., & Berthelson, M. (2011). Column-oriented database management systems: What do they deliver for analysts?. *European Journal of Management*, *11*(3), 138. 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=78131552\&site=eds-live`

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

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*.

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`