2017 8th International Conference on Information Technology (ICIT)

Contemporary Improvements of In-Memory Databases: A Survey

Jihad Najajreh
Palestine Polytechnic University
College of Graduate Studies
Hebron, Palestine
najajrah@gmail.com

Faisal Khamayseh
Palestine Polytechnic University
College of Information Technology and Computer Engineering
Hebron, Palestine
faisal@ppu.edu

***Abstract*- The high demand on increasing the amount of daily data that companies are dealing with, decreasing the cost of computer RAM and increasing of computer processing power, are all reasons that lead to a new type of databases called In-memory database (IMDB). IMDB now offers a very high speed big data management and processing of real time requests and responses by hosting the whole data in main memory to eliminate the hard disc I/O latency. In this paper, recent improvements in research on IMDB based on different dimensions have been studied, such as concurrency control, indexing, recovery and replication. Moreover, some common commercial and open source systems that have been used in large companies have been reviewed.**

I. INTRODUCTION

In-memory databases have recently become available solutions for large companies that require a high level of performance and processing of huge amount of data, as well as avoiding traditional hard disk I/O latency. Latency has become unacceptable for such companies that require real time response and service requests within seconds.

Recent studies have mentioned that we create 2.5 quintillion bytes of data every day, and about 90% of all the data in the world today was created in the past few years [1] [2]. Data is generated by smart phones, posts on social media, maps, sensors, blogs, machines, etc.

Nowadays, the competition between companies is for fast data retrieval and processing of huge amount of data since the data now is open and available almost everywhere, and the race is to utilize valuable resources becomes harder and harder, to give companies competitive advantages over others.

Real world applications that process this amount of data, considering the variety of structured and unstructured data, have become a persistent need, and in-memory databases offer a proper solution, because they help in decision making and strategic planning during analysis of big data and production of proper reports on spot.

This paper is structured as follows: we will highlight In-memory databases technology and types of computer memory in section two. Section three will draw on the difference between column-based and row-based data layouts. Section four will review the recent studies for IMDBs focusing on indexing, concurrency control, recovery and replication; in section five, will view some IMDB systems and their features.

II. IN-MEMORY DATABASES TECHNOLOGY

The procedure of processing Big Data fast and getting real time response and reports is very vital for enterprise companies when the latency costs millions of dollars. Recent studies have shown that one second delay costs Amazon $1.6 billion yearly; Google loses 8 million searches per day for one second delay [3]. It is obvious that this delay in application and data processing is no longer acceptable.

Accordingly, research in In-memory databases started from the early 80s [4] as a result of the enhancements and low cost of computer main memory. The main objective behind these research studies was to host the whole database in computer main memory for fast access and real time analysis.

A. TYPES OF COMPUTER MEMORY

Computer memory is a critical part of computing power used in PCs, laptops, servers, mobile and other devices. The development of main memory as shown in Figure 1 [5] identifies different types of storage compared with capacity, speed and cost.

B. VOLATILE MEMORY AND NON-VOLATILE MEMORY

Volatile memory is a computer memory that requires power on to maintain its contents; if the power is off, the contents are lost [6]. On the contrary, the Non-volatile memory keeps its contents when the power is off.

- VOLATILE MEMORY

RAM (Random Access Memory) is the main type of personal computer memory; RAM has 2 types: SRAM and DRAM; studies have shown that the average cost of one Gigabyte memory in 1980 was $ 6,328,125 and it dropped to $4.37 per Gigabyte in 2015 [7]; memory capacity and bandwidth have doubled every three years [4]. This development lead IT companies to design new In-memory database management system to handle big data.

978-1-5090-6332-1/17/$31.00 ©2017 IEEE

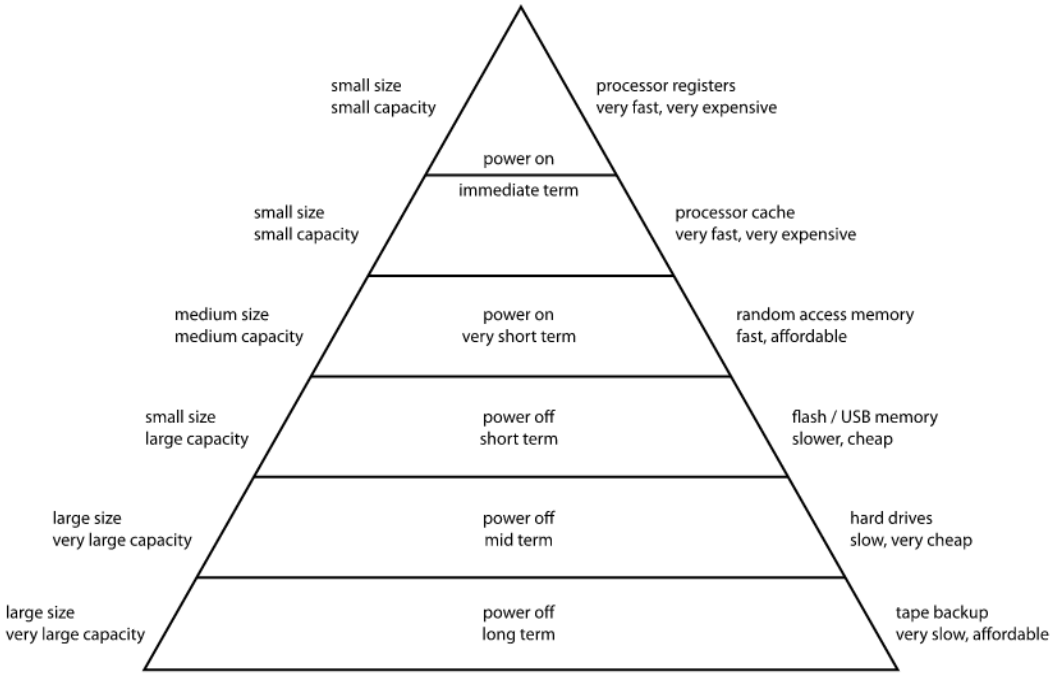


Figure 1: Computer Memory Hierarchy,
Retrieved from Wikipedia on 24 March 2016, from https://en.wikipedia.org/wiki/Memory_hierarchy

- NON-VOLATILE MEMORY

NVRAM and ROM are examples for Non-volatile memory used by emerging IT companies to host the whole database in NVRAM; it is slower than DRAM, but it gives more capacity with persistent storage if power is off [8], such as SSD (Solid-State Drive) and phase change memory (PCM).The following table compares the speed of different types of memory storage technologies [9]:

Table 1: Comparison of NVM technologies with other storage technologies

	DRAM	PCM	RRAM	MRAM	SSD	HDD
Read latency	60ns	50ns	100ns	20ns	25µs	10ms
Write Latency	60ns	150ns	100ns	20ns	300 µs	10ms
Volatile	Yes	No	No	No	No	No

Adapted from Arulraj, "Let's Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems," Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.

III. COLUMN-BASED VS. ROW-BASED IN-MEMORY DATA LAYOUT

Most of current IMDB systems use either, column-based or row-based for data store; for example, Facebook's Scuba database uses a row-based approach, and PowerDrill from Google uses column-based approach [10]; other systems use both row-based and columnar-base such as, Hekaton, SAP Hanna.

In a column-based approach, the system stores each column separately and contiguously; it is compressed on a separate location on disk or RAM, and it is usually used for analytical workloads systems that access large amounts of records with aggregate functions. Better Compression has been implemented on columnar data storage because the majority of the columns only store few distinct values [11].

Another advantage of columnar layout is that it increases the ability of parallel processing; operations on different columns can easily be processed in parallel since the data is vertically partitioned; for example, if the query needs to search or aggregate multiple columns, it can be divided into smaller tasks, and each task can be assigned to a different processor core and executed in parallel manner.

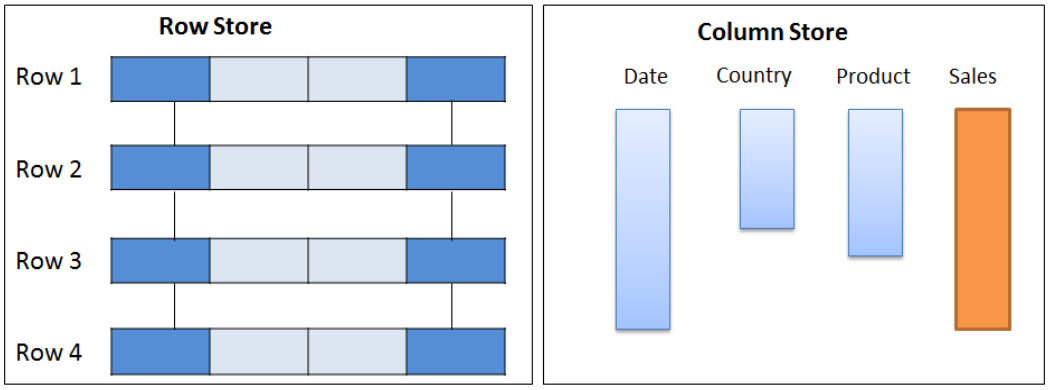
On contrary, row-based approach database systems store an entire record or the row one after the other contiguously in the storage media. For fast write operations, row-based layout is more suitable for transactional workloads that access many attributes on some records. Using both layouts (hybrid strategy), in hot and cold data, will enhance read/write and better query performance [10].

The following example shows the different usage of column-based and row-based storage [11]:

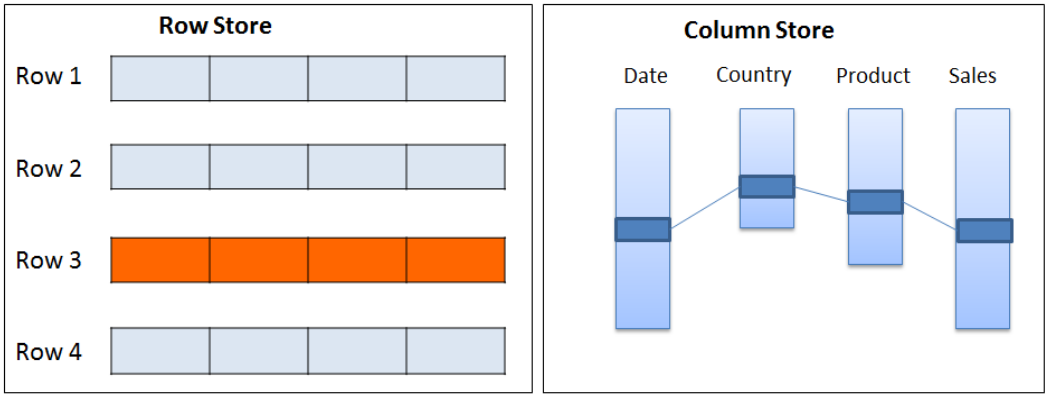
Table 2: Usage of column-based and row-based storage

	Date	Country	Product	Sales
Row 1	2013-01-01	India	Chocolate	1000
Row 2	2013-01-10	India	Ice-cream	2000
Row 3	2013-02-20	Germany	Chocolate	4000
Row 4	2013-03-01	US	Noodle	500

Column Operation: SELECT SUM(SALES) FROM SALES WHERE DATE > 2012-01-01



Row Operation: SELECT * FROM SALES WHERE COUNTRY = 'INDIA'



IV. Recent Improvements in IMDB

Recent research in in-memory databases handles different subjects and areas. In this section, we will review some improvements on IMDBs:

A. Indexing

Accessing data in main memory can be achieved by using tree structure indexing or hash functions. T-tree is a balanced tree type that is used specifically for in-memory databases [12]. IMDB indexes store just pointer to data not the data, because the collections of data exist in main memory.

Recent studies [4] [12] [13] [14] [15] have shown that there are many indexing techniques that have been implemented for IMDB such as: B+-Tree, wB+-tree, T-Tree, DCB-Tree, BD-Tree, FAST, HHB+tree and PI.

B+-Tree has been widely used in IMDB system [15] and the studies had shown that it had large extensive write overhead and CPU cache flush operations, and they proposed a new type of B+-tree called wB+-tree. Experimental results showed that the wB+-tree search speed has increased on DRAM and NVM.

Tree based indexing, has been modified and improved by [13] through saving bandwidth and storage using pruning technique to remove leaf nodes.

A T-tree is a balanced tree proposed for in-memory databases, where both the index and the actual data are fully kept in memory. T-trees nodes do not store the data; rather they just contain pointers to actual data fields [16].

DCB-Tree (Delta Cache B-tree) presented by [17] for an efficient main-memory index structure showed that the size of the index has been reduced to 80% at its best and by 30% in the worst case.

HHB+tree (hybrid hash index technique) is a proposed study by [18]. This technique minimizes split operations of overflow bucket using expanded hash index and also designed for NAND flash storage.

The newest study [19] for in-memory indexing proposed a new index structure, which is PI (Parallel In-memory Skip List Based Index). In this technique, requested queries are collected, and disjointly distributed for multiple threads for processing to avoid the use of latches.

B. Concurrency control

Multiple transactions on multiple servers and data scalability increase the need for efficient Concurrency Control mechanism. Several schemas are used in-memory databases. Many recent studies handle this challenge in IMDB systems [20] [21] [14] [22].

Multi-Version Concurrency Control (MVCC) is widely used. A comprehensive study in MVCC found in [9]. This omits the use of locking in database and enhances the performance by creating a new version of updated data and marks it as *latest* while the old data is marked as *obsolete* using timestamps. In this way, readers of data never block the writers of the same data.

In [23] a new extended version of MVCC proposed reducing overhead of tracking reads and eliminating validation phase, in order to achieve a better performance and to increase scalability.

Another improvement on MVCC is in [20]. The study proposed a new novel model for heavy and fast read transaction processing with little amount of snapshot isolation.

Another Concurrency Control schema has been implemented. It is called Optimistic Concurrency Control (OCC) [24] [25]. This method assumes that multiple transactions can be completed without interfering with each other. Before committing the data, each transaction verifies that no other transaction has modified the data it has been reading. If it checks conflicting modifications, the transaction will rollback and it can be restarted [26].

Other new Microsoft Research [27], have been achieved like Optimizing Optimistic Concurrency Control (OCC), by proposing new high performance algorithm to scale up the number of servers in multi core architecture.

A system that uses traditional CC schemas is also used in IMDB, because access time in main memory is faster than that of the hard disk. So, the locking will take less time. Strict timestamp ordering (STO) is a good and old technique where a recent study showed that STO has less impact on IMDB systems [28], and it is considered one of competitive CC techniques for CC.

C. In-Memory database recovery

Since IMDB exists in physical memory, and is distributed among multiple servers, the probability of system crash is rising. Backup must be in stable storage to ensure the durability of the system. The recovery process must ensure that backup is up-to-date and recovery mechanism procedure is to recover DB on failure.

To keep track of what is happening in DB, logging must be implemented. The logging techniques are used in IMDB in addition to traditional Write Ahead Logging (WAL) technique [29]. Log stream and checkpoint [14] [30] are also used to insure durability of DB when failure and recovery occurs.

REWIND is a modified version of Write Ahead mechanism accomplished in study [31]. The experimental results were of better performance by optimizing transaction log processing. A recent study

proposed a novel solution for recovery by using SCM (Storage Class Memory) [21].

SCM is Non-Volatile Memory, and its latency is close to DRAM plus durability and density. PCM Logging is another novel research proposed using PCM as media for login and recovery, because of its non-volatility, and low energy consumption [32].

Other studies proposed command logging which is log only operations, instead of log updated data [4].

Another recent study proposed a new Single-pass restore [33] by reducing the time and cost of restore and recovery and minimizing traditional log replay, by reorganizing the log into a different sort order while performing log archiving.

Table 2 summarizes the recent studies for IMDB systems.

Table 2: Comparison of IMDB systems and the contemporary improvements

Concurrency Control	IMDBM systems	Improvements	
MVCC	Oracle TimesTen, PostgreSQL, SAP HANA ,DBMS M, Microsoft Hekaton	Performance, Scalability	[20] [21] [14] [9] [22] [23]
OCC	Silo	Performance, Scalability	[24] [25] [27]
STO	HyPer		[28]
In-Memory database recovery			
WAL	BerkeleyDB ,Shore-MT, SQLite	Optimizing the transaction log processing	[31] [34]
SCM	any	Performance,	[21]
Single-pass restore	Shore-MT	Reducing the time and cost of media recovery	[33]
PCM	any	Minimize I/O accesses in disk-based, simplified recovery	[32] [32]
checkpoint	TimesTen, Microsoft Hekaton		[14] [35]
Indexing			
T-tree	Datablitz, Oracle TimesTen, EXtremeDB	Saving bandwidth and storage , new pruning tech.	[12] [13]
CSB+-Tree	NA	Performance	[36] [37]
Adaptive radix tree	HyPer		[38]
Bw-tree	Microsoft Hekaton		[14]
B+-Tree	H-store,		[15]
DCB-Tree	NA	Reduce Index size	[17]
PI	NA	Removing Lock	[19]
Hashing Index	Hekaton, Hyper, H-store, Redis		[14]

According to Table 2, it is obvious that improvements focused on optimizing memory indexing and managing concurrency controls, since the IMDBs are distributed for many clusters and need a special care in managing distributed requests.

Proposed new indexing techniques showed improvements in the performance and reduction of index size, such as Bw-tree, PI and hashing index with B+-trees.

Many modifications on concurrency control have been achieved to insure database durability and consistency. Many IMDB systems were adapted to use MVCC, because it is free lock for reader and writers.

Recovery in IMDB used different new and old techniques. Write A-head logging and checkpoints are old and are implemented in many Dbs. Some studies used PCM and SCM to write logging, because its characteristics are near or the same as RAM.

D. In-Memory database Replication

IMDB must guarantee high availability and scalability, since the number of users and machines has increased, and the data grows dramatically, by replicating the database to make secondary database on remote machines or other storage media. This will guarantee a fast recovery of database in case of a major issue happening on main database, and a distribution query processing among multiple cores or machines.

Google's Spanner database is scalable and multi-version globally distributed and synchronously replicated database. The data is automatically distributed and migrated on multiple nodes in data centers all over the world. It is designed to scale up to millions of machines and data centers [39]

MapReduce is a framework for processing and managing large amount of data distributed over thousands of nodes [40]. MapReduce is an example of replicated tasks not only the replication of data. In MapReduce, the job is divided into many small jobs; where each is assigned to different machines. SQL statements are replicated between these machines, such as replicated Joins [40].

V. IN-MEMORY DATABASE SYSTEMS

There are many commercial in-memory databases. Such in-memory databases include SQL server Hekaton, Oracle TimesTen, SAP Hana, VoltDB, SolidDB, IBM DB2 and more. Also, we have free open-source memory databases such as MonetDB, FastDB, H-Store, and Crescando.

The following figure has been created by [4]; it shows the common database management systems for HD, NVM and RAM.

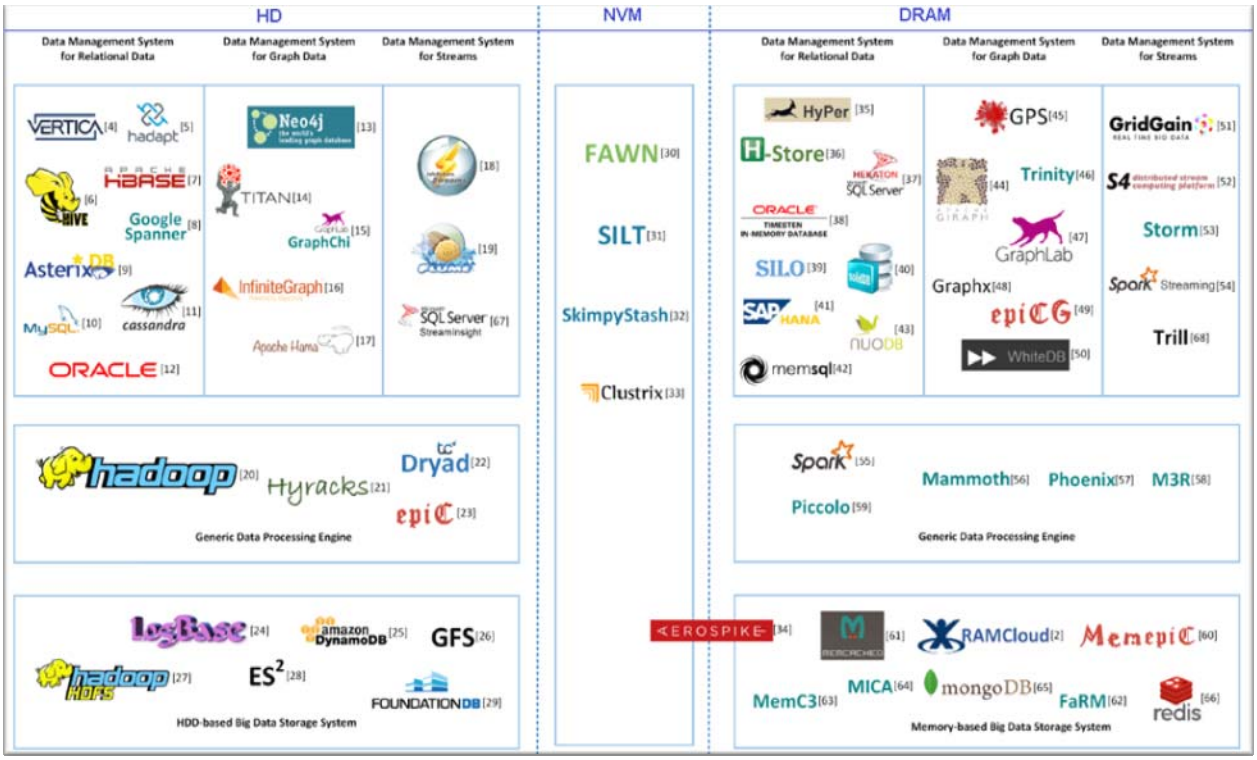


Figure 2: Database Management systems based on disk based or Memory-based
Source: H. Zhang, G. Chen, B. Chin Ooi, K.-L. Tan and M. Zhang, "In-Memory Big Data Management and Processing: A Survey," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, vol. 27, no. 7, pp. 1920-1948, 2015.

Figure 2 shows that there are many notable examples in industry for IMDB systems. In this section, we will present some features of the most common systems that have been widely used in large companies: SQL server Hekaton, SAP Hana, VoltDB, TimesTen.

A. SQL SERVER HEKATON

Hekaton is a major IMDB management system built on SQL server for OLTP workload. It is 100 times faster than the existing SQL Server and also used to work with memory-optimized tables and indexes and high numbers of processing cores with the ACID properties and free locks. In this management system, there are no data pages and it uses MVCC for concurrency control [41].

Hekaton was designed to manage and process frequently accessed data (hot data), and less frequently accessed data (Cold Data) that are automatically migrated to cheaper external storage, because it is not efficient, or sometimes impossible, to

keep all data in main memory in some cases. Hekaton has some limitations such as no foreign key constraints or check constraints [41].

B. SAP HANA (HIGH PERFORMANCE ANALYTIC APPLIANCE)

SAP Hana is an in-memory relational database management system that allows real time business intelligent response, supports relational, graphical and text data. SAP Hana stores data in both row-format and column-format to work with OLAP and OLTP systems.

The main characteristics of SAP Hana is that it provides access to real time analysis, increases speed of information processes such as planning, forecasting, and pricing and manages growing data volume and scalability. SAP Hana supports distribution across multiple hosts, where large tables can be partitioned to be processed in parallel. HANA database includes four dedicated servers IndexServe, Preprocessor is used for text processing; NameServer is responsible for locating data. Statistics Server is used to keep track performance parameters such as CPU usage, memory usage, etc.

C. VOLTDB

The key idea of VoltDb is partitioning the database tables across many independent sites and the stored procedures used to access these partitions. Multiple queries can be executed in parallel on each site independently [42]. A new node can be easily added to the database cluster to increase data capacity and to meet business and applications needs.

D. TIMESTEN

Oracle TimesTen is full featured relational in-memory database management system, storing all data in main memory and designed for business intelligence analytics to perform complex queries. TimesTen fully support semantic SQL transactions [43], high real-time throughput for enterprises and industries such as telecom, capital markets and defense [44].

VI. CONCLUSION

In-memory databases have become the solution for managing and processing big data, and generating real-time decisions based on large amounts of data, that the traditional DBMS cannot process. IMDB provides high performance business analytics and support transactional processing systems.

IMDBs host the whole data in computer memory instead of Hard disk. Replacing data layer enhances the performance thousands of times than that of traditional hard disk systems. This needs special indexing techniques, recovery, concurrency controls, and access methods. All of these areas are open subjects for research.

IMDB can offer dramatic improvements for enterprises and large companies, but this needs careful investigation to choose the proper provider, since each IDBM system has special characteristics and is suitable for different needs.

The new improvements have focused on Indexing and Concurrency Controls. A new novel method has been proposed such as a new extended version of MVCC. Bw+-tree, Hybrid hash index B-tree are proposed to minimize the size on index in main memories. OCC is also proposed by Microsoft with new optimization algorithm. IMDB recovery researches focused on WAL, REWIND. Some proposed to use SCM, PCM instead of main memory as storage media for writing the logs and recovery.

For future works, we recommend researchers to deeply focus on the following topics separately: IMDB garbage collector, memory allocation and deallocation, replications and storage management implementation techniques.

References

[1] IBM, "Storage Newsletter," 21 10 2011. [Online]. Available: http://www.storagenewsletter.com/rubriques/market-reportsresearch/ibm-cmo-study/. [Accessed 04 3 2016].

[2] M. Wall, " BBC News," BBC, 4 March 2014. [Online]. Available: http://www.bbc.com/news/business-26383058. [Accessed 4 3 2016].

[3] T. Bauer, "THE CONTEXT OF THINGS," 28 5 2014. [Online].Available: http://thecontextofthings.com/2014/06/28/amazon-and-impatience/. [Accessed 1 3 2016].

[4] H. Zhang, G. Chen, B. Chin Ooi, K.-L. Tan and M. Zhang, "In-Memory Big Data Management and Processing: A Survey," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,, vol. 27, no. 7, pp. 1920-1948, 2015.

[5] "Wikipedia," 5 March 2016. [Online]. Available: https://en.wikipedia.org/wiki/Memory_hierarchy. [Accessed 24 3 2016].

[6] "Wikipedia," 8 February 2016. [Online]. Available: https://en.wikipedia.org/wiki/Volatile_memory. [Accessed 24 March 2016].

[7] J. C. McCallum, "Statistics Brain Research Institute," Statistics Brain, 16th August 2015. [Online]. Available: http://www.statisticbrain.com/average-historic-price-of-ram/. [Accessed 24 March 2016].

[8] J. DeBrabant, J. Arulra, A. Pavlo, M. Stonebraker, S. Zdonik and S. R. Dulloor, "A prolegomenon on oltp database systems for non-volatile memory," Proceedings of the VLDB Endowment, vol. 7, no. 14, 2014.

[9] J. Arulraj, A. Pavlo and S. R. Dulloor, "Let's Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems," Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, pp. 707-722, 2015.

[10] A. Wen Jun Lu, "A study of an in-memory database system for real-time analytics on semi-structured data streams," Diss.University of Toronto, 2015.

[11] S. Hana, "saphanatutorial," saphanatutorial, [Online]. Available: http://saphanatutorial.com/column-data-storage-and-row-data-storage-sap-hana/. [Accessed 23 4 2016].

[12] R. BĂBEANU and M. CIOBANU, "In-memory databases and innovations in Business Intelligence," Database Systems Journal, vol. 1, 2015.

[13] T. Moataz1, T. Mayberry, E.-O. Blass and A. H. Chan, "Resizable tree-based oblivious RAM," Financial Cryptography and Data Security. Springer Berlin Heidelberg, pp. 147-167, 2015.

[14] C. Diaconu, C. Freedman, E. Ismert, P.-Å. Larson, P. Mittal, R. Stonecipher, N. Verma and M. Zwilling, "Hekaton: SQL Server's Memory-Optimized OLTP Engine," International Conference on Management of Data ACM, 2013.

[15] S. Chen and Q. Jin, "Persistent B+-Trees in Non-Volatile Main Memory," Proceedings of the VLDB Endowment, vol. 8, no. 7, 2015.

[16] "wikipedia.org," wikipedia.org, 22 1 2016. [Online]. Available: https://en.wikipedia.org/wiki/T-tree. [Accessed 17 04 2016].

[17] R. Binna, D. Pacher, T. Meindl and G. Specht, "The DCB-Tree: A Space-Efficient Delta Coded Cache Conscious B-Tree," In Memory Data Management and Analysis. Springer International Publishing, pp. 126-138, 2015.

[18] H. Ju and S. Cho, "HHB+tree Index for Functional Enhancement of NAND Flash," International Journal of Software Engineering and Its Applications, vol. Vol. 9, no. No. 9, pp. 289-294, 2015.

[19] Z. Xie, Q. Cai, B. Chin Ooi and W.-F. Wong, "PI : a Parallel in-memory skip list based Index," rXiv preprint arXiv, 2016.

[20] T. Neumann, T. Mühlbauer and A. Kemper, "Fast Serializable Multi-Version Concurrency Control for Main-Memory Database Systems," Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.

[21] I. Oukid, W. Lehner, T. Kissinger, T. Willhalm and P. Bumbulis, "Instant Recovery for Main-Memory Databases," CIDR, 2015.

[22] H. Mühe, "Concurrency in Main-Memory Database Systems," Diss. München, Technische Universität München, vol. Diss, 2014.

[23] J. Faleiro and D. Abadi, "Rethinking serializable multiversion concurrency control (Extended Version)," Proceedings of the VLDB Endowment 8.11, pp. 1190-1201, 2015.

[24] Y. Yuan, K. Wang, R. Lee, X. Ding, J. Xing, S. Blanas and X. Zhang, "BCC: Reducing False Aborts in Optimistic Concurrency Control with Low Cost for In-Memory Databases," Proceedings of the VLDB Endowment, p. 9.6, 2016.

[25] S. Tu, W. Zheng, E. Kohler, B. Liskov and S. Madden, "Speedy Transactions in Multicore In-Memory Databases," Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, 2013.

[26] "Wikipedia," Wikipedia, 5 April 2015. [Online]. Available: https://en.wikipedia.org/wiki/Optimistic_concurrency_control. [Accessed 01 April 2016].

[27] P. A. Bernstein, S. Das , B. Ding and M. Pilman, "Optimizing optimistic concurrency control for tree-structured, log-structured databases," Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.

[28] S. Wolf, H. M¨uhe, A. Kemper and T. Neumann, "An Evaluation of Strict Timestamp Ordering Concurrency Control for Main-Memory Database Systems," Springer International Publishing, pp. 82-93, 2015.

[29] N. Malviya, A. Weisberg, S. Madden and M. Stonebraker, "Rethinking Main Memory OLTP Recovery," Data Engineering (ICDE), 2014 IEEE 30th International Conference on. IEEE, 2014.

[30] L. Gruenwald, M. H: Dunham, J. Huang, J.-L. Lin and A. Chan Peltiery, "RECOVERY IN MAIN MEMORY DATABASES," 1996.

[31] A. Chatzistergiou, M. Cintra and S. D. Viglas, "REWIND: Recovery Write-Ahead System for In-Memory Non-," Proceedings of the VLDB Endowment (PVLDB),, vol. 8, no. 5, 2015.

[32] S. Gao, J. Xu, T. H¨arder, B. He, B. Choi and H. Hu, "PCMLogging: Optimizing Transaction Logging," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERINGS, pp. 3332-3346, 2015.

[33] C. Sauer, G. Graefe and T. H¨arder, "Single-pass restore after a media failure," BTW, pp. 217-236, 2015.

[34] W.-H. Kim, J. Kim, B. Nam and Y. Won, "NVWAL: Exploiting NVRAM in Write-Ahead Logging," Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems. ACM, pp. 385-398, 2016.

[35] H. Qian, "Efficient Durability Support for Multicore In-Memory Database," Efficient Durability Support for Multicore In-Memory Database, 2015.

[36] L. a. F. S. Wang, "Cost Analysis of B+-Tree and CSB+-Tree in Main Memory Database," 2015.

[37] L. Wang and F. Sun, "Cost Analysis of B+-Tree and CSB+-Tree in Main Memory Database," 3rd International Conference on Material, Mechanical and Manufacturing Engineering, 2015.

[38] U. Sirin, P. Tözün, D. Porobic and A. Ailamaki, "Micro-architectural Analysis of In-memory OLTP," SIGMOD No. EPFL-CONF-215922, 2016.

[39] Corbett, James C., et al., "Spanner: Google's globally distributed database," ACM Transactions on Computer Systems (TOCS), 2013.

[40] Li, Feng, et al., "Distributed data management using MapReduce," ACM Computing Surveys (CSUR), vol. 24, no. 3, 2014.

[41] K. Delaney, "simple-talk," simple-talk, 22 OCtober 2014. [Online]. Available: https://www.simple-talk.com/sql/database-administration/hekaton-in-1000-words/. [Accessed 19 April 2016].

[42] "VolDB," VolDB, 2016. [Online]. Available: https://docs.voltdb.com/UsingVoltDB/IntroHowVoltDBWorks.php. [Accessed 23 4 2016].

[43] C. S. Mullins, "techtarget," techtarget, June 2015. [Online]. Available: http://searchdatamanagement.techtarget.com/feature/Which-in-memory-DBMS-best-fits-your-companys-needs. [Accessed 23 4 2016].

[44] Oracle, "Oracle," Oracle, [Online]. Available: http://docs.oracle.com/cd/E21901_01/timesten.1122/e21631/overview.htm#TTCIN119. [Accessed 23 4 2016].