ISSN 2278-7763

Performance Management of In Memory Databases

By

Ram Babu, PHD Research Scholar – Bhagwant University,
Email : - singh.rambabu@gmail.com
Nirmal Lodhi, PHD Research Scholar – Bhagwant University,
Email : - nirmalrbsingh@gmail.com
Rajesh Pandey, PHD Research Scholar – Bhagwant University,
Email : - rajesh.fbd@gmail.com
R.C. Verma, Asst. Professor IMT Faridabad,
Email: - rcvranchi5@yahoo.co.in

Abstract:-

In-memory databases are becoming increasingly popular and an ever-more important factor in performance-critical activities such as stream processing and deep data analytics. Join Julian Stuhler as he delves into the world of in-memory databases: a technology that's both reassuringly familiar and intriguingly novel at the same time. In memory database systems are fastest database systems as the database itself resides in system's memory, so whenever there is any query comes to it need not to go to hard disk to read or write data to hard dis saving time and performance of the entire system.

Google, Twitter, Facebook and many others all rely on various forms of in-memory database to provide rapid response times in the face of ever-increasing data volumes.

Keywords: Memory, Database, Main memory

database, real time database, database management, Evolution, Mechanism, Architecture, Hybrid, Importance, Performance, Hard Disk, storage memory and products.

1. WHAT IS IN MEMORY DATABASE?

An in-memory database (IMDB) or main memory database system (MMDB) or real-time database (RTDB) is a database management system that primarily relies on main memory for computer data storage. It is contrasted with database management systems which employ a disk storage mechanism. Main memory databases are faster than disk-optimized databases since the internal optimization algorithms are simpler and execute fewer CPU instructions.
Accessing data in memory reduces the Disk seek when querying the data which provides faster and more predictable performance than disk.

2. EVOLUTION OF IN MEMORY DATABASES

I first bumped into the in-memory database concept back around 2001, and since then have considered it a classic disruptive technology that would inevitably marginalize, then unseat conventional SQL databases.

All the old R-series derived database optimization strategies from back in the 1970's operate in a linear fashion, like a zipper. They assume you read one record at a time from each table ("stream"), compare those records with each other ("join"), and immediately write the result back to disk

The old database design assumed you never have more than one record from each table in memory at a time, because who has that much memory? Everything is centered around that

ISSN 2278-7763

design assumption: you have a tiny amount of ram, so everything comes from disk in bite sized pieces and goes right back again. This assumption was undermined by Moore's Law, which doubled the available ram every year for the past few decades, and continues to do so.

In 2004, the department of defense commissioned a 2.5 terabyte ram disk for use as a database server. The industry switched to 64 bit (x86-64) processors in 2005. The Tyan Thunder supported 64 gigabytes of ram, the Nvidia MCP55 pro chipset supported 128 gigs, the elegantly named "Tyan S4989WG2NR-SI" supported 256 gigs, and these days the super micro 7500 supports half a terabyte of RAM.

So these days 64-bit systems with half a terabyte of ram are available retail. (And if you want to wander away from the PC, IBM PowerPC systems like the 780 server can hold a couple terabytes RAM each.)

Keeping all your tables in memory means you literally get a 3 orders of magnitude speedup (1000x), and you can use simple generic indexing strategies so the code becomes really simple. The first entirely in-memory database implementation I saw (a source forge project back in 2001) was a 1000 line python implementation on sourceforge that stored everything in python dictionaries. Over 90% of the code of that was implementing the SQL parser, not the actual database.

The more recent "nosql" movement is basically following up on this by saying "ok, if we've got our tables in memory why do we need SQL to talk to them?" The obvious way to design an in-memory database is thus in two parts: a direct layer providing function access to the data store with transactions and searching and persistence and such, and the other providing an SQL layer on top of that. Whether the database is in shared memory or not is an implementation detail.

The current crop of databases (mysql, postgresql, oracle, and so on) is all painfully obsolete. As with the introduction of UNIX, they were obsoleted by something much simpler, which figured out that 90% of the stuff they spent their time doing didn't need to be done at all.

3. MECHANISMS

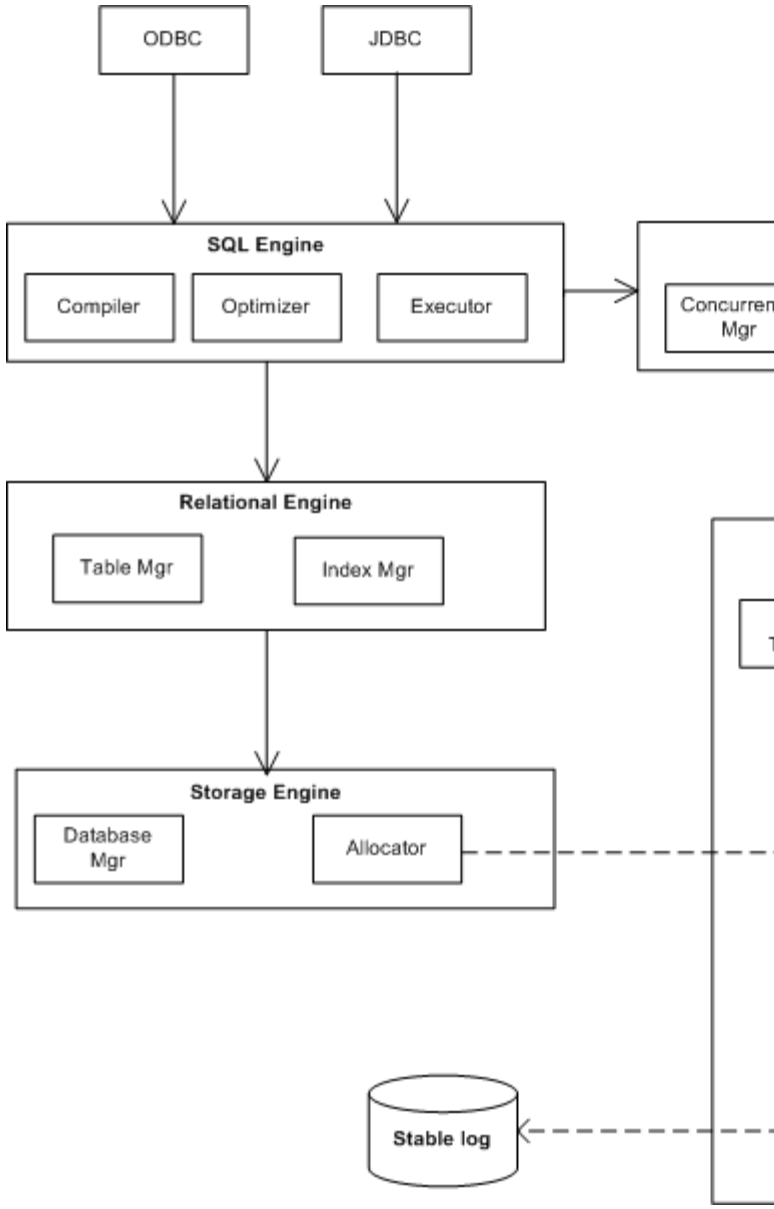
Many MMDBs add durability via the following mechanisms:
Snapshot files, or, checkpoint images, which record the state of the database at a given moment in time. These are typically generated periodically, or, at least when the MMDB does a controlled shut-down. While they give a measure of persistence to the data (in that not everything is lost in the case of a system crash) they only offer partial durability (as 'recent' changes will be lost). For full durability, they will need to be supplemented by one of the following:

Transaction logging, which records changes to the database in a journal file and facilitates automatic recovery of an in-memory database

High availability implementations that rely on database replication, with automatic failover to an identical standby database in the event of primary database failure To protect against loss of data in the case of a complete system crash, replication of a MMDB is normally used in conjunction with one or more of the mechanisms listed above.
Some MMDBs allow the database schema to specify different durability requirements for selected areas of the database - thus, faster-changing data that can easily be regenerated or that has no meaning after a system shut-down would not need to be journal for durability (though it would have to be replicated for high availability), whereas configuration information would be flagged as needing preservation.

ISSN 2278-7763

4. ARCHITECTURE



Figure, depicts a main memory database management system. This has nearly all the components, which are present in disk resident database management system. Implementations of components under SQL Engine, Relational Engine, and Storage Engine differ heavily from the DRDB components.

ISSN 2278-7763

5. HYBRIDS WITH ON-DISK DATABASES

The first database engine to support both in-memory and on-disk tables in a single database were released in 2003. The advantage to this approach is flexibility: the developer can strike a balance between performance (which is enhanced by sorting, storing and retrieving specified data entirely in memory, rather than going to disk); cost, because a less costly hard disk can be substituted for more memory; persistence; and form factor, because RAM chips cannot approach the density of a small hard drive.

Manufacturing efficiency is another reason a combined in-memory / on-disk database system may be chosen. Some device product lines, especially in consumer electronics, include some units with permanent storage, and others that rely on memory for storage (set-top boxes, for example). If such devices require a database system, a manufacturer can adopt a hybrid database system at lower and upper cost, and with less code customization, than using separate in-memory and on-disk databases, respectively, for its disk-less and disk-based products.

6. STORAGE MEMORY

Another variation is to have large amounts of nonvolatile memory in the server. For example Flash memory chips as addressable memory rather than structured as disk arrays. A database in this form of memory combines very fast access speed with persistence over reboots and power losses.

7. IMPORTANCE OF IN MEMORY DATABASE

In applications where response time is critical, such as telecommunications network equipment and mobile advertising networks, main memory databases are often used. IMDBs have gained a lot of traction, especially in the Data analytics space, starting mid-2000s mainly due to cheaper RAMs.

In addition to providing extremely fast query response times, in-memory analytics can reduce or eliminate the need for data indexing and storing pre-aggregated data in OLAP cubes or aggregate tables. This capacity reduces IT costs and allows faster implementation of BI / BA applications.

Three developments in recent years have made in-memory analytics increasingly feasible: 64-bit computing, multi-core servers and lower RAM prices

Main memory databases store data on volatile memory devices. These devices lose all stored information when the device loses power or is reset. In this case, MMDBs can be said to lack support for the durability portion of the ACID (atomicity, consistency, isolation, durability) properties. Volatile memory-based MMDBs can, and often do, support the other three ACID properties of atomicity, consistency and isolation.

8. PRODUCTS

I first bumped into the in-memory database concept back around 2001, and since then have considered it a classic disruptive technology that would inevitably marginalize, then unseat conventional SQL databases

In actual, development of a relational in-memory database system was started at Perihelion Software in 1991, and had its first commercial release in early 1993 under the name Polyhedra. The product was later spun out as a separate company, which was acquired by Enea AB in 2001. Polyhedra were developed from the start as a commercial offering for use in SCADA and embedded systems.

ISSN 2278-7763

Companies needing a fast data storage mechanism for their own products have often developed their own, in-house solution which they later marketed commercially. For example, research in main-memory database systems started around 1993 at Bell Labs. It was prototyped as the Dali Main-Memory Storage Manager.[8] This research lead to the commercial main-memory database, Datablitz.

In 2001, McObject introduced eXtremeDB, the first in-memory database system targeting embedded systems, with early adoption in sectors including telecom, industrial control and consumer electronics. Later, they added features associated with non-embedded computing, including 64-bit support and SQL / ODBC / JDBC interfaces, and eXtremeDB has seen adoption in enterprise, financial and Web-based systems requiring low latency.

In recent years, main memory databases have attracted the interest of larger database vendors.

Ehcache, is an in-memory database created by the developers of Terracotta, Inc.. It can hold the largest amount of data in memory on the smallest number of servers. SAG acquired the San Francisco based company in 2010 to imbed with their BPM solutions.

TimesTen, a start-up company founded by Marie-Anne Neimat in 1996 as a spin-off from Hewlett-Packard, was acquired by Oracle Corporation in 2005. Oracle now markets this product as both a standalone database and an in-memory database cache to the Oracle database.

In 1999, Altibase Corporation developed an In-Memory DBMS offering. In 2012, Altibase released version 6 of its hybrid database flagship product, ALTIBASE HDB.

Also in 1999, Microsoft COM+ IMDB solution provided an application with fast access to data through OLE DB, without incurring the overhead associated with storing and accessing data to and from physical disks that worked within Windows NT 3.5 and then upcoming Windows 2000.

IBM acquired solidDB in 2008, and Microsoft is widely rumored to be launching an in-memory solution in 2009. VoltDB, founded by DBMS pioneer Michael Stonebraker, announced the general availability of its namesake in-memory database in May 2010, and offers versions of the product under open source (GPLv3) and commercial licenses.

SAP AG announced general availability of their own in-memory product, SAP HANA, in June 2011.

Oracle also has a similar product to SAP's called Oracle Exalytics.

WebDNA 7, released as freeware, is a robust hybrid in-memory database system and scripting language designed for the World Wide Web.

9. SUMMARY

Finally, no discussion on in-memory databases would be complete without at least a brief mention of the high performance in-memory database systems that have been built from the ground up to support today's search and social networking sites. Google, Twitter, Facebook and many others all rely on various forms of in-memory database to provide rapid response times in the face of ever-increasing data volumes. Other data in memory solutions such as SolidDB (acquired by IBM in 2007) provide a somewhat more generalized solution to address many of the same issues, and can even be used as a kind of high performance front end to more conventional disk-based databases.

From a simple HDD swap to SSD-aware RDBMS systems and beyond, in-memory databases are becoming increasingly popular and an ever-more important factor in performance-critical activities

ISSN 2278-7763

such as stream processing and deep data analytics. Expect to hear much more about them over the next few years.

10. ACKNOWLEDGMENT

My sincere thanks to Dr. A.K. Yadav to his encouragement and guidance in writing research papers on importance and performance of large database management systems which in turn allowed me to focus on In Memory database systems and their performance.

REFERENCES:-

1. In Memory Database definition, WhatIs.com
2. Evolution: landley.net.
3. Julian Stuhler, databasejournal.com on IBM DB2 article
4. wikipedia.org and wikimedia.org