Rise of NewSQL

Hemang Tailor, Sushant Choudhary, Vinay Jain

[1]Computer Science Department, K.J. Somaiya College of Engineering, Mumbai, Maharashtra, India
Hemang.t@somaiya.edu
[2]Computer Science Department, K.J. Somaiya College of Engineering, Mumbai, Maharashtra, India
sushant.c@somaiya.edu
[3]Computer Science Department, K.J. Somaiya College of Engineering, Mumbai, Maharashtra, India
vinay.jain@somaiya.edu

ABSTRACT

This paper describes the special features of NewSQL, Features like storing of data, security, support to big data etc. NewSql will overcome all constraints present in SQL and NoSQL. As it becomes ever more pervasively engaged in data driven commerce, a modern enterprise becomes increasingly dependent upon reliable and high speed transaction services. At the same time it aspires to capitalize upon large inflows of information to draw timely business insights and improve business results. These two imperatives are frequently in conflict because of the widely divergent strategies that must be pursued :the need to bolster on-line transactional processing generally drives a business towards a small cluster of high-end servers running a mature, ACID compliant, SQL relational database; while high throughput analytics on massive and growing volumes of data favor the selection of very large clusters running nontraditional (NoSQL/NewSql) databases that employ softer consistency protocols for performance and availability.

Keywords — Databases, SQL, NoSQL, NewSQL, ACID Properties

1. INTRODUCTION

A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies. Database management systems (DBMSs) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, LibreOffice Base and FileMaker Pro. A database is not generally portable across different DBMSs, but different DBMSs can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one database. Formally, "database" refers to the data themselves and supporting data structures. Databases are created to operate large quantities of information by inputting, storing, retrieving and managing that information. Databases are set up so that one set of software programs provides all users with access to all the data.

A "database management system" (DBMS) is a suite of computer software providing the interface between users and a database or databases. Because they are so closely related, the term "database" when used casually often refers to both a DBMS and the data it manipulates. Outside the world of professional information technology, the term database is sometimes used casually to refer to any collection of data (perhaps a spreadsheet, maybe even a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system. The interactions catered for by most existing DBMSs fall into four main groups:

Data definition – Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.

Update – Inserting, modifying, and deleting data.

Retrieval – Obtaining information either for end-user queries and reports or for processing by applications.

Administration – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information if the system fails.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails. Both a database and its DBMS conform to the principles of a particular database model. "Database system" refers collectively to the database model, database management system, and database.

Physically, database servers are dedicated computers that hold the actual databases and run only the DBMS and related software.

Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage. RAID is used for recovery of data if any of the disks fail. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database applications.

2. INTRODUCTION TO NEW SQL

NewSQL is a new database access language. It is easier to learn than SQL, elegant, consistent, and well defined. It is not an extension or subset of SQL, and not an Object database language. It is based on top of the cross database library LDBC.SQL is outdated, too complex and has many flaws. Object databases are not the future. Existing applications and databases are supported. Two types of converters will be built: SQL > NewSQL and NewSQL > SQL. (SQL means all major SQL dialects). That not only helps migration, it also allows to run existing applications with different databases. Here are some first ideas. The functionality of the language will be standardized.

3. PAGE STYLE

In this section NewSql is compared with Traditional sql and NoSQL. This Comparison is based on the various parameters and that parameters are discussed below and the open source databases are considered here like MySQL for Sal, Mongo DB for NoSQL and NuoDb, VoltDb for NewSql.

The parameters are listed below

3.1. Support to ACID Properties

In computer science, ACID (Atomicity, Consistency, Isolation, and Durability) is a set of properties that guarantee that database transactions are processed reliably. In the context of databases, a single logical operation on the data is called a transaction. For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction.

Atomicity: Atomicity (database systems)

Atomicity requires that each transaction is "all or nothing": if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors, and crashes. To the outside world, a committed transaction appears (by its effects on the database) to be indivisible ("atomic"), and an aborted transaction does not happen.

Consistency: Consistency (database systems)

The consistency property ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules, including but not limited to constraints, cascades, triggers, and any combination thereof. This does not guarantee correctness of the transaction in all ways the application programmer might have wanted (that is the responsibility of application-level code) but merely that any programming errors do not violate any defined rules.

Isolation: Isolation (database systems) the isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially, i.e. one after the other. Providing isolation is the main goal of concurrency control. Depending on concurrency control method, the effects of an incomplete transaction might not even be visible to another transaction.[

Durability: Durability (database systems)

Durability means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. In a relational database, for instance, once a group of SQL statements execute, the results need to be stored permanently (even if the database crashes immediately

thereafter). To defend against power loss, transactions (or their effects) must be recorded in a non-volatile memory.

The Traditional sql supports the ACID properties. All the 4 property are supported by sql. NoSQL does not support the ACID properties. NoSQL supports BASE properties that is [Basic Availability Soft-state Eventual consistency].NewSql supports the ACID properties.

3.2. Storage

In SQL data is stored in RDBMS. In NOSQL data is stored in graphs and trees and NewSql gives both the advantages. IN SQL goes in one machine where as In NoSQL data is distributed and NEWSQL gives advantages of both.

Fig-1: Comparison of storage in NoSQL and NewSQL

3.3. CAP Theorem

Brewer's 2000 talk was based on his theoretical work at UC Berkley and observations from running Inktomi, though Brewer and others were talking about trade-off decisions that need to be made in highly scalable systems years before that (e.g. "Cluster-Based Scalable Network Services" from SOSP in 1997 and "Harvest, yield, and scalable tolerant systems" in 1999) so the contents of the presentation weren't new and, like many of these ideas, they were the work of many smart people (as I am sure Brewer himself would be quick to point out).The three requirements are: Consistency, Availability and Partition Tolerance, giving Brewer's Theorem its other name - CAP.

Consistency: A service that is consistent operates fully or not at all. Gilbert and Lynch use the word "atomic" instead of consistent in their proof, which makes more sense technically because, strictly speaking, consistent is the C in ACID as applied to the ideal properties of database transactions and means that data will never be persisted that breaks certain pre-set constraints. But if you consider it a preset constraint of distributed systems that multiple values for the same piece of data are not allowed then I think the leak in the abstraction is plugged (plus, if Brewer had used the word atomic, it would be called the AAP theorem and we'd all be in hospital every time we tried to pronounce it.

Availability: Availability means just that - the service is available (to operate fully or not as above). When you buy the book you want to get a response, not some browser message about the web site being uncommunicative. Gilbert & Lynch in their proof of CAP Theorem make the good point that availability most often deserts you when you need it most - sites tend to go down at busy periods precisely because they are busy. A service that's available but not being accessed is of no benefit to anyone.

Partition Tolerance: If your application and database runs on one box then (ignoring scale issues and assuming all your code is perfect) your server acts as a kind of atomic processor in that it either works or doesn't (i.e. if it has crashed it's not available, but it won't cause data inconsistency either).

The Significance of the Theorem CAP Theorem comes to life as an application scales. At low transactional volumes, small latencies to allow databases to get consistent has no noticeable effect on either overall performance or the user experience. Any load distribution you do undertake, therefore, is likely to be for systems management reasons. But as activity increases, these pinch-points in throughput will begin limit growth and create errors. It's one thing having to wait for a web page to come back with a response and another experience altogether to enter your credit card details to be met with "HTTP 500 java.lang.schrodinger.purchasingerror" and wonder whether you've just paid for something you won't get, not paid at all, or maybe the error is immaterial to this transaction. Who knows? You are unlikely to continue, more likely to shop elsewhere, and very likely to phone your bank. The database crowd, unsurprisingly, like database technology and will tend

to address scale by talking of things like optimistic locking and sharding), keeping the database at the heart of things.

NoSQL: In recent years NoSQL databases have justified providing eventual or other weak read/write consistency as an inevitable consequence of Brewer's CAP Theorem. This relies on the simplistic interpretation that because distributed systems can't avoid Partitions, they have to give up Consistency in order to offer Availability. This reasoning if flawed - CAP does allow systems to maintain both Consistency and Availability during the (majority of the time) when there is no Partition, and good distributed systems strive to maximize both C and A while accounting for P. You do not have to give up consistency as a whole just to gain scalability and availability.

NewSQL: NewSQL actually cracks the CAP theorem in NewSQL all 3 things are satisfied but problem of latency occurs sometimes that is problem of processing speed.

Of equations as if they appeared directly in the text.

3.4 Performance:

The following diagram shows the performance of the NewSQL.

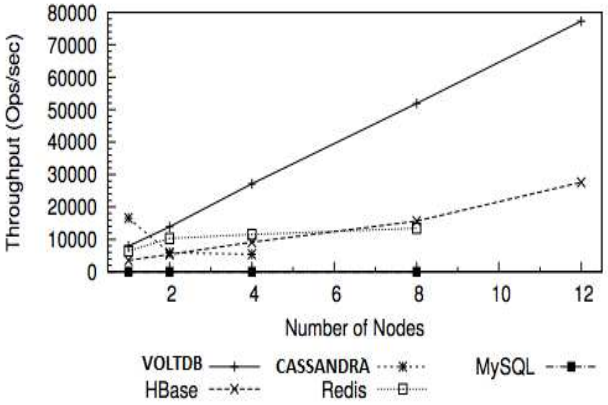

Fig-2: Performance of NewSQL

3.5 Consistency of Data:

The Following diagram shows the consistency of the data. For the traditional SQL smaller data range the performance is quite good but when data size increases at that time problem arises same case with the NOSQL but NewSQL is better than both the SQL. Performance of NewSQL quite good the following diagram shows that quite well if data is in GB Traditional sql is perfect and For tera bytes of data which is real need NewSQL is perfect but in penta bytes of data consistency of NoSQL decreases the performance of system.

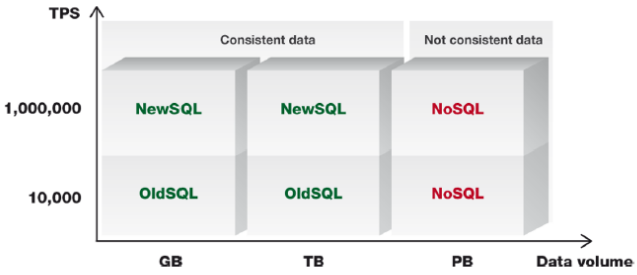

Fig-3: Data Consistency in NewSQL

3.6 Security:

Database security concerns the use of a broad range of information security controls to protect databases (potentially including the data, the database applications or stored functions, the database systems, the database servers and the associated network links) against compromises of their confidentiality, integrity and availability. It involves various types or categories of controls, such as technical, procedural/administrative and physical. Database security is a specialist topic within the broader realms of computer security, information security and risk management.

Traditionally databases have been largely secured against hackers through network security measures such as firewalls, and network-based intrusion detection systems. While network security controls remain valuable in this regard, securing the database systems themselves, and the programs/functions and data within them, has arguably become more critical as networks are increasingly opened to wider access, in particular access from the Internet. Furthermore, system, program, function and data access controls, along with the associated user identification, authentication and rights management functions, have always been important to limit and in some cases log the activities of authorized users and administrators.

NOSQL doesn't support ACID supports and security constraint is there. Problem of authentication, authorization and confidentiality. But all above drawbacks are covered In NEWSQL that is authentication, authorization and confidentiality are better in NEWSQL.

3.7 In Memory Database

An in-memory database (IMDB; also main memory database system or MMDB or memory resident database) is a database

management system that primarily relies on main memory for computer data storage. It is contrasted with database management systems that 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 eliminates seek time when querying the data, which provides faster and more predictable performance than disk. 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 RAM. With the introduction of NVDIMM technology, in-memory databases will now be able to run at full speed and maintain data in the event of power failure.

Traditional Sql doesn't support this special feature that is in memory databases. NOSQL and NEWSQL both supports this special feature.

3.8 Big Data

Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, duration, storage, search, sharing, transfer, analysis and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions.". A visualization created by IBM of Wikipedia edits. At multiple terabytes in size, the text and images of Wikipedia are a classic example of big data.

As of 2012, limits on the size of data sets that are feasible to process in a reasonable amount of time were on the order of Exabyte's of data. Scientists regularly encounter limitations due to large data sets in many areas, including meteorology, genomics, connectomics, complex physics simulations, and biological and environmental research. The limitations also affect Internet search, finance and business informatics. Data sets grow in size in part because they are increasingly being gathered by ubiquitous information-sensing mobile devices, aerial sensory technologies (remote sensing), software logs, cameras, microphones, radio-frequency identification readers, and wireless sensor networks. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012, every day 2.5 Exabyte's (2.5×1018) of data were created. The challenge for large enterprises is determining who should own big data initiatives that straddle the entire organization.

Big data is difficult to work with using most relational database management systems and desktop statistics and visualization packages, requiring instead "massively parallel software running on tens, hundreds, or even thousands of servers". What is considered "big data" varies depending on the capabilities of the organization managing the set, and on the capabilities of the applications that are traditionally used to process and analyze the data set in its domain. "For some organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data size becomes a significant consideration".

SQL don't support this big data but NOSQL and NEWSQL both supports big data.

3. 9. OLTP

Online transaction processing, or OLTP, is a class of information systems that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing. The term is somewhat ambiguous; some understand a "transaction" in the context of computer or database transactions, while others (such as the Transaction Processing Performance Council) define it in terms of business or commercial transactions. OLTP has also been used to refer to processing in which the system responds immediately to user requests. An automated teller machine (ATM) for a bank is an example of a commercial transaction processing application.

Online transaction processing increasingly requires support for transactions that span a network and may include more than one company. For this reason, new online transaction processing software uses client or server processing and brokering software that allows transactions to run on different computer platforms in a network. In large applications,

efficient OLTP may depend on sophisticated transaction management software (such as CICS) and/or database optimization tactics to facilitate the processing of large numbers of concurrent updates to an OLTP-oriented database.

For even more demanding decentralized database systems, OLTP brokering programs can distribute transaction processing among multiple computers on a network. OLTP is often integrated into service-oriented architecture (SOA) and Web services. Online Transaction Processing (OLTP) involves gathering input information, processing the information and updating existing information to reflect the gathered and processed information. As of today, most organizations use a database management system to support OLTP. OLTP is carried in a client server system SQL is to slow for OLTP and does not scale according to the requirement. NOSQL problem is lack of consistency and low-level interface but NEWSQL comes good here. NewSQL is fast, scalable and consistent and supports SQL.

	Old SQL	NoSQL	NewSQL
Relational	Yes	No	Yes
SQL	Yes	No	Yes
ACID transactions	Yes	No	Yes
Horizontal scalability	No	Yes	Yes
Performance / big volume	No	Yes	Yes
Schema-less	No	Yes	No

Table-1: Summary

4. APPLICATION: GOOGLE SPANNER

Spanner is a scalable, globally-distributed database designed, built, and deployed at Google. At the highest. Level of abstraction, it is a database that shards data across many sets of Paxos state machines in datacenters spread all over the world. Replication is used for global availability and geographic locality; clients automatically failover between replicas. Spanner automatically reshards data across machines as the amount of data or the number of servers changes, and it automatically migrates data across machines (even across datacenters) to balance load and in response to failures. Spanner is designed to scale up to millions of machines across hundreds of datacenters and trillions of database rows.

Applications can use Spanner for high availability, even in the face of wide-area natural disasters, by replicating their data within or even across continents. Our initial customer was F1, a rewrite of Google's advertising backend. F1 uses five replicas spread across the United States. Most other applications will probably replicate their data across 3 to 5 datacenters in one geographic region, but with relatively independent failure modes. That is, most applications will choose lower latency over higher availability, as long as they can survive 1 or 2 datacenter failures.

Spanner's main focus is managing cross-datacenter Replicated data, but we have also spent a great deal of time in designing and implementing important database features on top of our distributed-systems infrastructure. Even though many projects happily use bitable, we have also consistently received complaints from users that bitable can be difficult to use for some kinds of applications: Those that have complex, evolving schemas, or those that want strong consistency in the presence of wide-area replication. (Similar claims have been made by other authors) Many applications at Google have chosen to use Megastore because of its semi relational.

Data model and support for synchronous replication,

Despite its relatively poor write throughput. As a consequence, Spanner has evolved from a bitable-like versioned key-value store into a temporal multi-version database. Data is stored in schematized semi-relational tables; data is versioned, and each version is automatically time stamped with its commit time; old versions of data are subject to configurable garbage-collection policies; and applications can read data at old timestamps. Spanner supports general-purpose transactions, and provides a SQL-based query language. Google Spanner is based on NEWSQL.

5. CONCLUSIONS

The NewSql is next big thing in the field of databases. As day by day changing world we need both conventional as well as new way of storing the data that is the relational databases and NoSQL we need advantages of both the technologies that can be achieved through the NewSQL. From the above comparisons we can see that there is significant amount of difference in all the 3 technologies and NewSql provides better way of storing and maintaining the data.

REFERENCES

[1] Aslett, Matthew "How Will The Database Incumbents Respond To NoSQL And NewSQL?". Group-451,2012.

[2] Stonebraker, Michael "NewSQL: An Alternative to NoSQL and Old SQL for New OLTP Apps". Communications of the ACM Blog. ,2012

[3] Hoff, Todd . "Google Spanner's Most Surprising Revelation: NoSQL is Out and NewSQL is In". 2012

[4] Lloyd, Alex, "Building Spanner". Berlin Buzzwords ,2012

[5] Cattell, R. . "Scalable SQL and NoSQL data stores". ACM SIGMOD Record 39 (4): 12. doi:10.1145/1978915.1978919

[6] Aslett, Matthew. "Is H-Store the future of database management systems?" ,2008

[7] Dignan, Larry . "H-Store: Complete destruction of the old DBMS order?",2012

[8] Venkatesh, Prasanna . "NewSQL - The New Way to Handle Big Data" ,2014

