NewSQL Databases
MemSQL and VoltDB Experimental Evaluation

João Oliveira[1] and Jorge Bernardino[1,2]
[1]*ISEC, Polytechnic of Coimbra, Rua Pedro Nunes, Coimbra, Portugal*
[2]*CISUC – Centre for Informatics and Systems of the University of Coimbra, Coimbra, Portugal*

Keywords: NewSQL Databases, NewSQL, MemSQL, VoltDB, TPC-H.

Abstract: NewSQL databases are a set of new relational databases that provide better performance than the existing systems, while maintaining the use of the SQL language. Due to the huge amounts of data stored by organizations these types of databases are suitable to process efficiently this information. In this paper, we describe and test two of the most popular NewSQL databases: MemSQL and VoltDB. We show the advantages of the NewSQL databases engines using the TPC-H benchmark. The experimental evaluation demonstrated the ability of MemSQL and VoltDB to execute effectively TPC-H benchmark queries.

1 INTRODUCTION

NewSQL is a set of new SQL databases engines with high-performance and scalability. These engines seek to promote the same performance and scalability improvement of NoSQL systems, designed solutions that have the advantages of the relational model, and with the benefit of using SQL language (Stonebraker, 2012).

The term NewSQL was first used by analyst Matthew Aslett in 2011 in this "NoSQL, NewSQL and Beyond" (Aslett, 2011) business analysis report, which discussed the emergence of new databases systems.

NewSQL aims to provide the same performance as NoSQL systems for OLTP loads and still maintain the ACID (Atomicity, Consistency, Isolation, Durability) guarantees of traditional databases.

Matthew Aslett says NewSQL databases are designed to meet the scalability requirements of distributed architectures or to improve performance such that horizontal scalability is no longer a necessity, including new storage mechanisms, transparent shadowing technologies, and databases completely new.

The relational databases technology was invented in 1970 by Edgar Frank Codd, where he demonstrated the functionalities of this technology. Simultaneously was developed the SQL language that has become the standard language for manipulation of relational model. SQL is a complete language, used both to create and to manage, update, retrieve, or share information. Relational databases have been a common choice for storing information since the 1980s. Despite all the advantages, SQL also has many limitations both at the level of databases management, such as scalability, performance and size. SQL databases are losing processing power with the increase in data volume. The main problem is the existence of differences in language that allows access to the databases, because not all vendors use the standard in their entirety. The existence of differences in the syntax rules in each product, makes databases difficult to use. So, it can be said that NewSQL aims to achieve high performance and have great ability to resize, and intends to preserve SQL. There are two characteristics that are common, supporting the relational data model and the use of SQL as the main interface (Pavlo and Aslett, 2016).

In this paper, we intend to study and test NewSQL databases, for the experimental evaluation we considered the MemSQL and VoltDB engines, using the TPC-H benchmark.

The rest of this paper is structured as follows. The next section presents related work. Section 3 describes MemSQL and VoltDB NewSQL databases. Section 4 describes the TPC-H benchmark used. Section 5 presents the configuration used for testing followed by experimental evaluation and results. Finally, section 6 presents the conclusions and future work.

Oliveira J. and Bernardino J.
NewSQL Databases - MemSQL and VoltDB Experimental Evaluation.
DOI: 10.5220/0006518902760281
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KEOD 2017)*, pages 276-281
ISBN: 978-989-758-272-1

2 RELATED WORK

In this section we describe some approaches, functionalities, comparisons of relational databases, NoSQL and NewSQL related to our work.

Pavlo and Aslett (2016) discussed the emergence of NewSQL databases, providing a detailed explanation of the term NewSQL, and their characteristics. More specifically, they studied the new DBMS constructed from scratch. Instead of focusing on an existing system, they started from a database without any architectural approach, representing a development of database technologies that incorporates existing ideas into unique platforms that form new engines, in a new era in which computing resources are abundant and accessible.

In (Moniruzzaman, 2014), the authors discusses the NewSQL data management system and compares with NoSQL and with the traditional database system. This article discusses the architecture, characteristics, and classification of NewSQL databases for online transaction processing (OLTP) for managing large data.

In (Grolinger et al., 2013) the authors analyse NoSQL and NewSQL solutions with the goal of providing guidance to professionals. They also give a survey to choose the appropriate storage of data, and identity challenges and opportunities, scaling data storage are investigated, partitioning, replication, consistency, and concurrency control. In addition, the use cases and scenarios in which the NoSQL an NewSQL data stores were used are discussed and the suitability of various solutions for different sets of applications are examined.

In (Kumar et al., 2014) the authors include the basic concept of large data and its benefits, as well as the data types, and the introduction to Apache Hadoop. In addition, this article contains the introduction of NoSQL, NewSQL as well as its features and analyses how to handle data through Hadoop, NoSQL and NewSQL.

Binani, Gutti and Upadhyay (2016) describe an approach of SQL databases also known as RDBMS (Relational Database Management Systems) to meet the needs of big data systems, which are mainly unstructured in nature and expect quick response and scalability. NoSQL databases are also analysed to provide scalability and a structured platform for big data applications. However, due to some disadvantages, NewSQL comes up, it is a relational database with scalability properties. This paper discusses each of these database systems and tries to solve problems of data requirements.

In (Lourenco *et al.*, 2015a), the authors used a real world enterprise system with real corporate data to evaluate the performance characteristics of popular NoSQL databases and compare them to SQL counterparts. They tested Cassandra, MongoDB, Couchbase Server and MS SQL Server databases and compared their performance while handling demanding and large recording write requests from a real company data with an electrical measurement enterprise system.

In (Lourenço *et al.*, 2015b), the authors make a concise and up-to-date comparison of NoSQL engines. The most beneficial use case scenarios from the software engineer´s point of view, their advantages and drawbacks by surveying the currently available literature were described.

In our work, the focus is to conduct an experimental evaluation using two popular NewSQL engines, more specifically testing the execution time of TPC-H benchmark queries.

3 NEWSQL DATABASES

In this section we describe two of the most popular NewSQL databases engines: MemSQL and VoltDB.

3.1 MemSQL

MemSQL is a distributed, in-memory, relational database management system (RDBMS), which comply with structured query language (SQL).

MemSQL uses a two-tiered architecture consisting of aggregator nodes and leaf nodes. Aggregator nodes are cluster-aware query routers that act as a gateway into the distributed system. They store only metadata and reference data. Aggregators intelligently distribute queries across the leaf nodes and aggregate results that are sent to the client. Increasing the number of aggregators will improve operations like data loading and will allow MemSQL to process more client requests concurrently.

Leaf nodes function as storage and compute nodes. Data is automatically distributed across leaf nodes into partitions to enable parallelized query execution. Increasing the number of leaf nodes will increase the overall capacity of the cluster and speed up query execution, especially queries that require large table scans and aggregations. Additional leaf nodes also allow the cluster to process more queries in parallel. The number of aggregator and leaf nodes deployed determines cluster capacity and performance. Figure 1 shows MemSQL architecture.

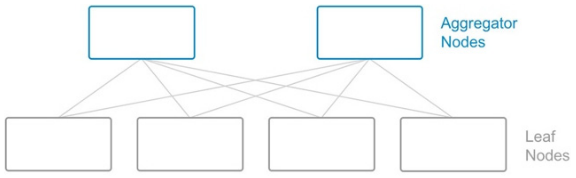


Figure 1: MemSQL Architecture (MemSQL Documentation, 2017).

Allocation of data is through two types, replicate tables, and distributed tables. Replicated tables are copied to all nodes, and fragmented nodes are distributed through fragmentation keys, which facilitates the execution of join operations. To fragment a table, MemSQL supports both primary and derived fragmentation. Primary fragmentation can be accomplished by any attribute of the tables that is specified as a fragmentation key. Derived fragmentation is a strategy to try to allocate fragments that have the same key value in the same node.

For concurrency control, MemSQL uses the MVCC protocol in conjunction with non-blocking indexes to ensure better performance in scenarios with concurrent operations compromising consistency, so read operations do not block writes (Chen et al., 2016).

MemSQL can dynamically add nodes to increase storage capacity or processing power. Queries are compiled and converting into C++ code, then stored in a cache. The code is reusable, but the cache does not store results of executions from previous queries. In addition to redundancy, MemSQL ensures durability with logging and full database snapshots. If a node goes down, its state can be recreated by replaying the most recent snapshot and log files. (Pavlo and Aslett, 2016).

In Figure 2, the query is received, MemSQL checks whether there is already hashed code, if the parameters are passed to the already compiled code the query is executed, otherwise the query will be compiled and stored in the hash table. In Table 1 we present MemSQL properties.

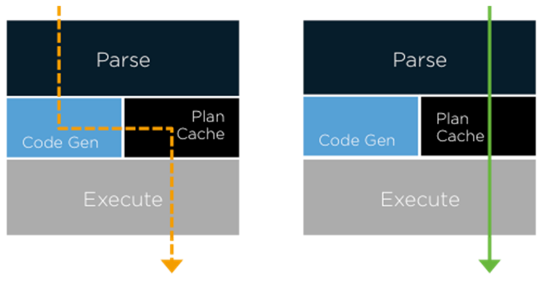


Figure 2: Query execution in MemSQL (MemSQL Documentation, 2017).

Table 1: MemSQL features (DB Engines, 2017).

	MemSQL
Memory Storage	Yes
Partitioning	Yes
Concurrency Control	MVCC
Replication	Strong + Passive
Development language	C++

3.2 VoltDB

VoltDB is an in-memory database, which depends on the main memory for data storage. This system was designed in 2010 by the well-known database researchers, Michael Stonebraker, Sam Madden, and Daniel Abadi (Stonebraker, 2012).

VoltDB is an ACID relational database that uses a shared-nothing architecture, ensuring that the data is always correct and available. The data is organized into memory partitions, and transactions are sent by clients connected to the database.

VoltDB uses horizontal scalability to increase the capacity of the nodes of the existing database, or the number of nodes in a shared-nothing cluster.

For high availability VoltDB uses partitions which are transparently replicated across multiple servers. If one fails all data remains available and consistent for continuum operation. Memory performance with durability on the disk is possible with the VoltDB snapshot. The snapshot is a complete copy of the database at a certain point in time that is written on the disk.

VoltDB uses asynchronous replication on the WAN (Wide Area Network) for loss recovery. The remote copy is a read-only while it is not considered to be the primary database.

In Figure 3 we present the architecture of VoltDB and in Table 2 its main features (Stonebraker, 2012).

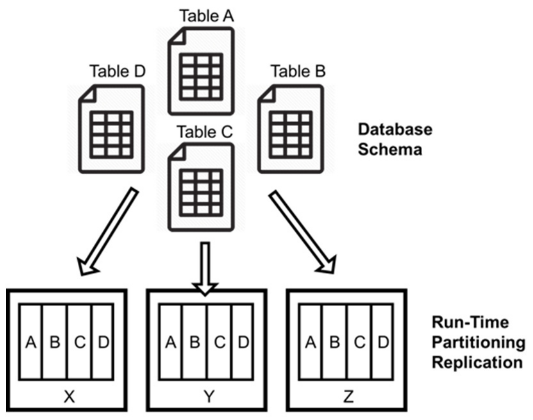


Figure 3: VoltDB Architecture (VoltDB Documentation, 2017).

Table 2: VoltDB features (DBEngines, 2017).

	VoltDB
Memory Storage	Yes
Partitioning	Yes
Concurrency Control	Yes
Replication	Strong + Passive
Development language	Java, C++

4 TPC-H BENCHMARK

TPC-H is a decision-support benchmark consisting of a set of business-oriented *ad-hoc* queries. This benchmark evaluates the performance of various decision support types by performing a set of controlled queries on the databases under test. These queries are much more complex than most OLTP transaction and include a wide set of operators and selectivity constraints. The purpose of this benchmark is to reduce the diversity of operations found in an information analysis application, while retaining the application's essential performance characteristics, namely: the level of system utilization and the complexity of operations. As example, we show below Q1 that is a pricing summary report query. This query reports the amount of business that was billed, shipped, and returned The pricing summary report query provides a summary pricing report for all lineitems shipped as of a given date:

```
select l_returnflag, l_linestatus,
sum(l_quantity) as sum_qty,
sum(l_extendedprice) as sum_base_price,
sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price, sum(l_extendedprice * (1 -
l_discount) * (1 + l_tax)) as sum_charge,
avg(l_quantity) as avg_qty,
avg(l_extendedprice) as avg_price,
avg(l_discount) as avg_disc, count(*) as
count_order
from LINEITEM where l_shipdate <= '1998-12-
01' - interval '117' day
group by l_returnflag, l_linestatus
order by l_returnflag, l_linestatus;
```

The TPC-H is a data model having eight tables as we can see in figure 4, which involves customers, suppliers and purchases of items. Five continents are represented, which contains twenty-five nations, represented by the Region and Nation tables. Customers and suppliers are stored in the tables, Supplier and Customer, which are associated with the nations. The Orders table places purchase orders. There is the Partsupp table to record the relationship between vendors and items. Finally, there is the more bulky table, Lineitem that associates purchase items with orders (Santos et al., 2011).

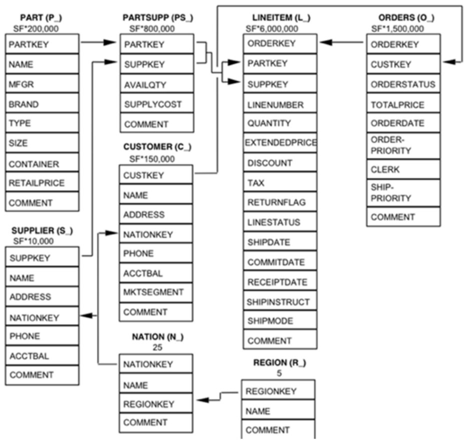


Figure 4: TPC-H benchmark schema.

Table 3 presents TPC-H eight tables as well as the number of records that each table has for 1GB.

Table 3: Number of records for TPC-H with 1GB.

Tables	#Records for 1GB
Nation	25
Region	5
Part	200 000
Supplier	10 000
Partsupp	800 000
Customer	150 000
Orders	1 500 000
Lineitem	6 001 215
TOTAL	***8 661 245***

5 EXPERIMENTAL EVALUATION

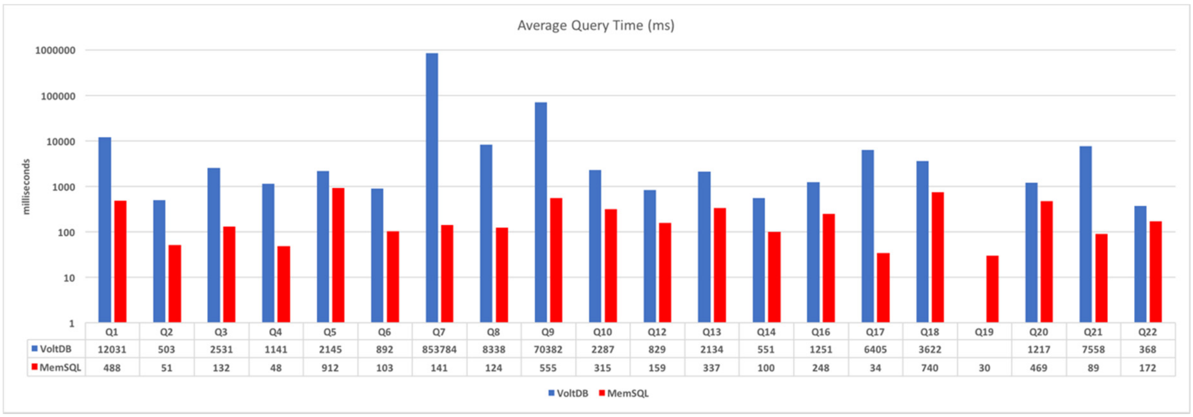
In the next sections, we will give information about the experimental setup. It is described the characteristics of computers used, the load times of the tables in the MemSQL and VoltDB engines, the execution time of the queries. Finally, we will discuss the results obtained in the experimental evaluation.

5.1 Experimental Setup

We evaluate MemSQL and VoltDB database with the TPC-H benchmark queries using a scale factor of 1.

The tests were performed on a computer with the following characteristics:

- Operating system Ubuntu 16.04 64-bit LTS
- Intel Xeon (R) CPU X5570 @ 2.93GHz x16, 40GB memory (RAM) and 250GB disk.



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q12	Q13	Q14	Q16	Q17	Q18	Q19	Q20	Q21	Q22
🟦 VoltDB	12031	503	2531	1141	2145	892	853784	8338	70382	2287	829	2134	551	1251	6405	3622		1217	7558	368
🟥 MemSQL	488	51	132	48	912	103	141	124	555	315	159	337	100	248	34	740	30	469	89	172

Figure 5: Average query execution time of TPC-H queries.

5.2 Loading Information

In this section, we present the results of the time in the insertion of data obtained from each of the eight tables in the TPC-H benchmark. The results are for loading data using the two NewSQL engines, MemSQL and VoltDB. The loading time was obtained on the computer with the specifications mentioned in the setup of the experimental setup section.

Table 4 below shows the insertion time in seconds.

Table 4: Loading times for 1GB of data.

Tables	VoltDB	MemSQL
Nation	0.32 sec	0.17 sec
Region	0.48 sec	0.18 sec
Part	9.64 sec	0.65 sec
Supplier	0.89 sec	0.35 sec
Partsupp	32.56 sec	0.86 sec
Customer	7.14 sec	0.49 sec
Orders	68.74 sec	2.90 sec
Lineitem	339.71 sec	33.58 sec

From the table above, it can be verified that the Lineitem table is the one that took the most time to load the data in the two systems due to having 6 001 215 rows and the 25 rows Nation table was the least time consuming to insert the data.

Summing up all the loading times for each of NewSQL databases it can be verified that the value obtained to VoltDB is 459.48 seconds and only 39.18 seconds to MemSQL, meaning that it is 11.7 times faster.

5.3 Results and Discussion

The experimental evaluation was performed with a TPC-H scale factor of 1, which corresponds to a 1 GB data size. The tests were performed through five executions of the 20 queries sequentially, so we intend to avoid the effects of caching, and the final value shown in the results is the average value.

In this experimental evaluation, it was not possible to execute all the 22 queries of TPC-H. Query Q11 could not be executed in the NewSQL engines because it did not support the *Having* condition, and query Q15 because it is not possible to create views. Query Q19 in VoltDB did not produce any results, after several hours of execution, we cancel it execution; this is because VoltDB uses a large quantity of memory (RAM) to perform the searches.

In Figure 5, we present the average query execution time obtained for TPC-H queries. The time presented is expressed in milliseconds, with logarithmic scale of base 10.

When analyzing the results of the queries it is verified that MemSQL execution time, are smaller compared to those of the VoltDB.

The queries, Q1, Q8, Q17, Q19, Q21 are those that have higher execution times in VoltDB, since they search information in a larger number of tables and need to do more computations to present the result. The remaining ones', search information in a smaller number of tables and in a row limit, which leads to the search for a smaller number of information, therefore, a shorter execution time.

Queries, Q1, Q5, Q9, Q18 and Q20 have a longer execution time in MemSQL, for the same reason mentioned for the VoltDB, they search information in a larger number of tables and the remaining queries in a smaller number of tables with limitation of lines that explain a shorter execution time.

When analyzing the execution times of the queries used, it was verified that the queries take more time in the VoltDB engine compared to the MemSQL.

Longer searches look for information in the TPC-H tables with the largest number of information (Lineitem, Orders, Customer, Supplier). It is understandable that the search time is longer when it is necessary to look up information in the tables with largest number of rows. We also verify that queries with aggregations have the slowest execution time.

Based on the experimental results, it is possible to verify that MemSQL stands out, with a 92% improvement over VoltDB. As mentioned in section 3.1 in MemSQL, when a new search is received on the system, it checks if there is already a hash code, then it reuses the previously compiled code, passing the parameters to the already compiled code, significantly reducing the processing time. Due to this distinct feature, the query execution time of MemSQL queries are smaller compared to VoltDB.

In this way it was verified that the two NewSQL engines are fast to present the results of the searches but the MemSQL was able to surpass the VoltDB in all the searches as we can verify in Figure 5. The VoltDB being a system based on the memory, uses the memory on the computer considerably, and consequently led to longer execution times.

6 CONCLUSIONS AND FUTURE WORK

Today information is vital for organizations that have multiple data sources and systems to store them. However, there is a huge problem due to the massive quantity of data inserted in the databases, that causes poor query performance and worse data analysis.

These problems highlighted the advantage of NewSQL databases by providing increased throughput, and improved performance, solving also storage problems. With this it is possible to solve the current storage problems, but also fix some flaws that exist in other database systems, that is why NewSQL databases are designed to be scalable and support large amounts of data and remain efficient.

During the evaluation of the tests, only Q19 was not possible to execute. Therefore, it is possible to verify that NewSQL systems use considerably the primary memory to perform searches. To obtain good performance it is necessary to have a computer with good processing and storage capacity that contributes to the better results of NewSQL databases.

We can conclude that MemSQL NewSQL engine behaved better than the VoltDB for 1GB of data. Moreover, MemSQL use standard SQL, without the necessity of queries rewriting, while in VoltDB it is necessary to rewrite the queries. For example, VoltDB does not support the date type, but only the timestamp date type.

As a future work, we intend to evaluate other NewSQL database engines and comparing the performance with traditional relational databases, such as MySQL or PostgreSQL. We also intend to increase the scale factor of TPC-H using a database size of 10GB and more.

REFERENCES

Aslett, Matthew, 2011. *NoSQL, NewSQL and Beyond*, https://blogs.the451group.com/information_management/2011/04/15/nosql-NewSQL-and-beyond/

Binani, S., Gutti, A. and Upadhyay, S. (2016) 'SQL vs. NoSQL vs. NewSQL-A Comparative Study', *Communications on Applied Electronics*, 6(1), pp. 43–46.

Chen, J. *et al.* (2016) 'The MemSQL Query Optimizer : A modern optimizer for real-time analytics in a distributed database', 9(13), pp. 1401–1412. doi: 10.14778/3007263.3007277.

DBEngines (2017). [online] Available at https://db-engines.com/en/ranking [Accessed 7 Jun. 2017].

Grolinger, K. *et al.* (2013) 'Data management in cloud environments: NoSQL and NewSQL data stores', *Journal of Cloud Computing: Advances, Systems and Applications*, 2, p. 22.

Kumar, R. *et al.* (2014) 'Apache Hadoop, NoSQL and NewSQL Solutions of Big Data', *International Journal of Advance Foundation and Research in Science & Engineering*, 1(6), pp. 28–36.

Lourenço, J.R. *et al.* (2015a) 'Choosing the right NoSQL database for the job: a quality attribute evaluation', Journal of Big Data, 2 (1), art. no. 18.

Lourenço, J. R., *et al.* (2015b) 'NOSQL databases: A software engineering perspective', *Springer Advances in Intelligent Systems and Computing*, 353(6), pp. 741–750.

Moniruzzaman, A. B. M. (2014) 'NewSQL : Towards Next-Generation Scalable RDBMS for Online Transaction Processing (OLTP) for Big Data Management', *International Journal of Database Theory & Application*, 7(6), pp. 121–130.

Pavlo, A. and Aslett, M. (2016) 'What's Really New with NewSQL?', *SIGMOD Record*, 45(2), pp. 45–55. doi: 10.1145/3003665.3003674.

Santos, R. J., Bernardino, J. and Vieira, M. (2011) 'Balancing security and performance for enhancing data privacy in data warehouses', Proc. 10th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, TrustCom 2011, pp. 242-249.

Stonebraker, M. (2012) 'New opportunities for New SQL', *Communications of the ACM*, 55(11), p. 10. doi: 10.1145/2366316.2366319.

TPC-H Documentation. (2017, July) Retrevied from http://www.tpc.org/tpch/default.asp

VoltDB Documentation. (2017, July) Retrevied from https://www.voltdb.com.