INFO-H-415

Advanced Databases

Project

NewSQL Databases and NuoDB

Ibrahim TOURE
Issam Hajji

December 18, 2017

Contents

1 Introduction

In the past few years, The term NewSQL has seen a rise in popularity since it was used in a 2011 paper discussing the rise of new database systems as challengers to established vendor[1].
What exactly do NewSQL databases offer that other current Databases like NoSQL or classic Relational SQL databases do not offer ? We will provide a description of NewSQL databases, discuss their underlying features by addressing the flaws of NoSQL systems and the limitation of SQL systems.

Secondly, we will study a NewSQL database called NuoDB and showcase it's important features, it's architecture and how it has implement the NewSQL features.

Finally, we will present our experiment with NuoDB describing our use case, the environment in which we made the experiment and the results of the experiment.

2 NewSQL databases

2.1 Relational database

A Database Management System is a software for sharing and storing data in a database. These databases consist of a table set connected to each other.

To ensure the proper functioning of a database, the DBMS will maintain several properties that are referred to as ACID (atomicity, consistency, isolation, durability).

• Atomicity : This property will ensure that all of the requests in the transaction will complete, or none will, and the database state is left unchanged.

• Consistency : Ensure the fact that any transaction will bring the database from one valid state to another

• Isolation : Mechanisms that allow the management of competitive access to database, no transaction interferes with another transaction.

• Durability : This property will ensure that once a transaction has been committed, it will remain so.

[1] Aslett, Matthew (2010). "What we talk about when we talk about NewSQL". 451 Group (published 2011-04-06). Retrieved 2012-10-07.

Respecting these properties keeps the database in a consistent state, but it makes it unable to handle very large volumes of data at extreme data rates and increases latency due to mechanisms such as locks or logging.

The relational DBMS will therefore show their limit when they are confronted with very high data rates as well as the management of data types that are not compatible with the relational model schemas.

2.2 Flaws of NoSQL databases

NoSQL Databases are non-relational database technologies that offer high performance. As opposed to traditional relational databases that use table to store data (schemas), they do not enforce the use of a schema. The retrieval of data is done by using a partition key (or hash key). The type of data can be columns sets, or structured documents like XML or JSON.
The flexible data model of NoSQL databases like DynamoDB (especially in a cloud) offer the ability to scale horizontally.
However, NoSQL database technologies present serious flaws. Most importantly, they break the ACID (Atomicity, Consistency, Isolation, Durability) (see) guarantees. The ACID guarantees enables the use of transactions against a Database.

With the lack of ACID guarantees, a great number of transaction-oriented applications (OLTP [2]) are not feasible with NoSQL databases.

One example of a typical OLTP oriented application is the *Twitter* social network. Users submitting tweets or a modifying their profiles imply a great number of transactions against a database (INSERT, UPDATE, DELETE). These transactions are typically short but great in number (number of users is massive).
These type of applications must use technologies that not only offer ACID guarantees but also offer the same scalability and high performance than NoSQL databases.

[2]OLTP : Online Transaction Processing. Online transaction processing (OLTP) is information systems that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing.

2.3 What are newSQL databases ?

NewSQL is a new class of relational database management systems for online transaction processing that provide the same scalable performance as NoSQL for read and write workloads while using the SQL language and maintaining the ACID properties that ensure the consistency of stored data.

This system differs from relational DBMS by the use of several noSQL features such as:

• Column oriented data storage : Allows you to add columns more easily to tables without having to resize rows, and also column compression, which is more efficient than row compression when the data in the column is similar.

• Processing In-memory : Instead of storing the data in hard disks, and transferring them to a cache memory during transactions, some NewSQL database will use the cache as the primary storage for storing, managing, and manipulating data. This significantly improves the performance of the requests.

• Multiprocessing : The newSQL uses a parallel architecture, such as SMP, to increase the computing power of the servers by using multiple processors that share a single memory.

• Lock-free concurrency control : A mechanism that avoids locks during concurrent read operations with writes. Real-time playback is thus facilitated.

Traditional databases cannot provide a response to the demand when a large number of users are reached because the work required to expand the database is substantial. To overcome the problem of scalability, the developers add scaling technologies such as partitioning, sharding and clustering or change servers to achieve a higher performance, which is often quite expensive.

The NewSQL DBMS can elastically scale which allows to add a new machine to a running database and to make it operational immediately. More information about distribution databases are given in section 3.

2.4 What problem do they solve ?

As we have seen in the previous section, NewSQL databases offers relational data, guarantees "acidity" and also offers the scalability and high performance of NoSQL

databases. OLTP intensive applications will benefit from NewSQL databases as they will not only be guaranteed acidity but will also scale nicely.
Traditionally, if an organization needed a larger database and better performance, they would need to invest in a more powerful and bigger server. However, using a NewSQL database nullifies this need as it can scale horizontally (add new servers to cluster).

Additionally, NewSQL databases address the problem with database specific data querying in NoSQL databases as they comply to the SQL standard. Application are thus easier to migrate from a traditional SQL database to a new SQL database.

3 NuoDB: an elastic NewSQL database

3.1 Features

The NuoDB database technology offers the following features :

- ACID-Compliant,
- Elastic SQL: NuoDB maintains SQL as it scales up,
- Elastic Scalability: Adding/removing capacity on demand without compromising acidity,
- Active-active: Always on and Always available, .

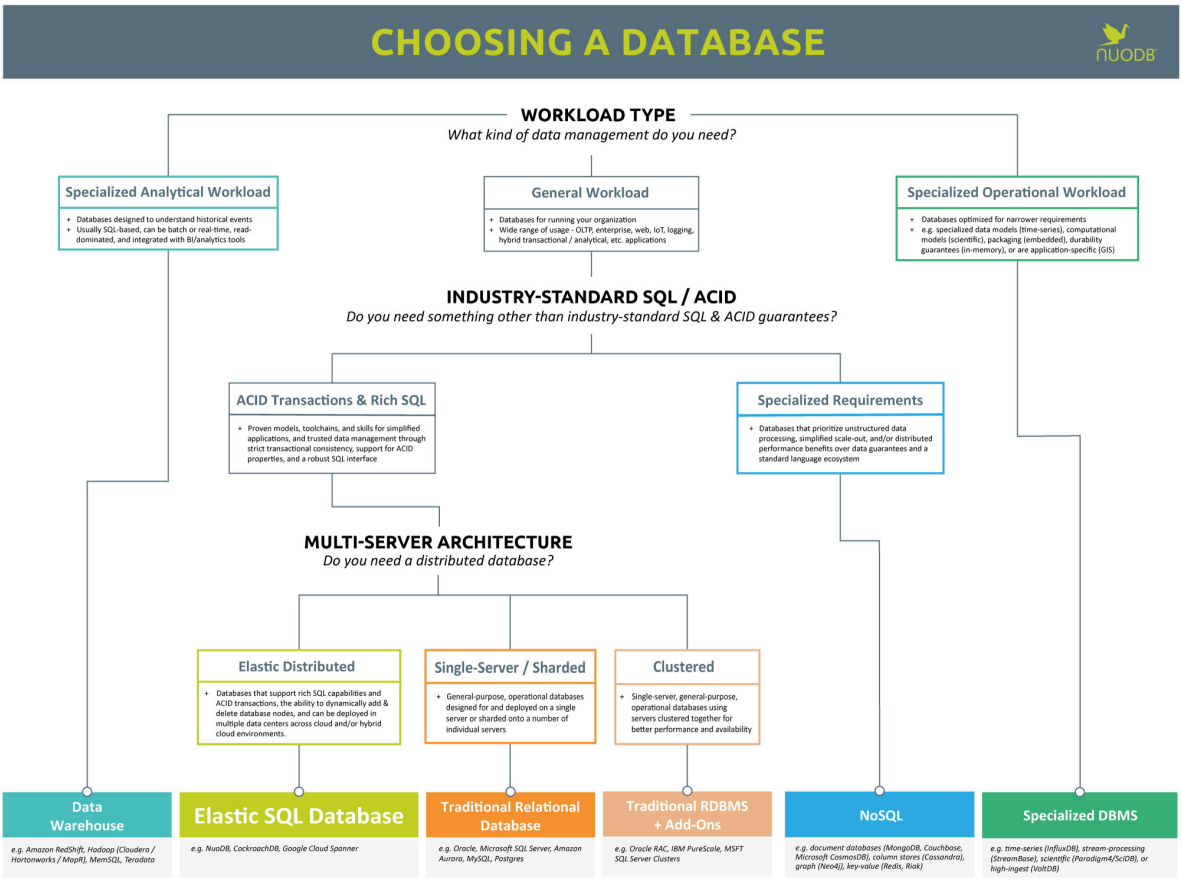

Figure 1: Choosing a database - Decision Tree

The decision tree in the figure above shows that NuoDB is useful for applications with general workload that need ACID and Rich SQL compliance and must offer scalability (adding capacity on demand).

3.2 Architecture

3.2.1 Two Tiers

NuoDB is a distributed architecture split into two layers: a transactional tier and a storage tier.
The scaling is made possible by the use of these two tiers. One tier is responsible for durability (the storage tier) and the other one is responsible for transactions (transactional tier).
Because of this separation between responsibilities it is possible for NuoDB to scale : increasing throughput can be achieved by adding an instance of either tiers without

compromising the data. They also handle failures independently, hence why a crash is not fatal.

Note that the NuoDB architecture also has a third tier that is responsible for administration (NuoDB administration). An overview of the architecture can be seen in the following figure :

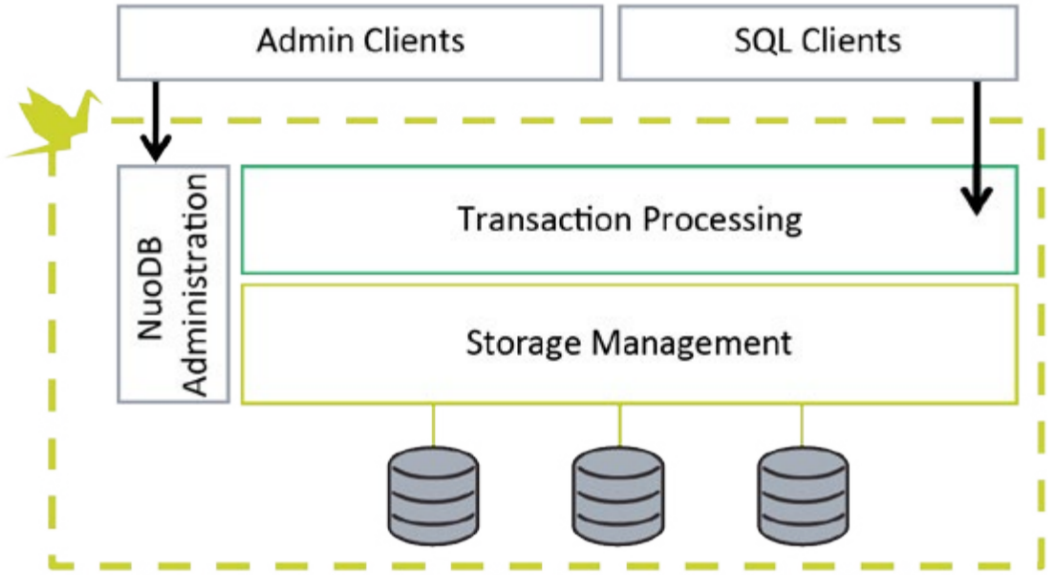

Figure 2: 2 Tier Architecture

The **transaction layer** is responsible for maintaining Atomicity, Consistency, and Isolation in running transactions. It has no visibility into how data is being made durable. It is a purely in-memory tier, so it's efficient as it has no connection to durability. The transactional tier is an always-active, always consistent, on-demand cache.

The **storage management** tier ensures Durability. It's responsible for making data durable on commit and providing access to data when there's a miss in the transactional cache. It does this through a set of peer-to-peer coordination messages.

3.2.2 Peer-to-Peer communication & Scaling

The two tiers discussed above comprise a set of processes that are of two types : **Storage managers** (SM) or **transaction engines** (TE). The architecture is thus a peer-to-peer

network. The peers communicate with each other with the SRP protocol [3].

TEs handle the SQL client connections and run SQL queries against cached data. Two scenarios can occur : Either the data is in the cache, the TE returns that data or there is a cache miss (the data is not present in the cache). In that case, the TE will contact another TE or a SM for that data. At any time t, TEs know which peers is fastest by constantly running a cost function. In theory, duplicating TEs will boost performance by a factor of 2. The setup process for a newly created TE is fast. The TE must simply authenticate and setup a few objects and is then ready to accept connections.

SMs handle the persistence of data, they receive data from TEs and store that data or send data to a TE that has made a request.
Achieving minimal redundancy is as simple as configuring two SMs. A new SM can be added at any time. It will synchronize with the other SMs and keep an updated copy of the database.
In the following figure can be seen an example of a configuration:

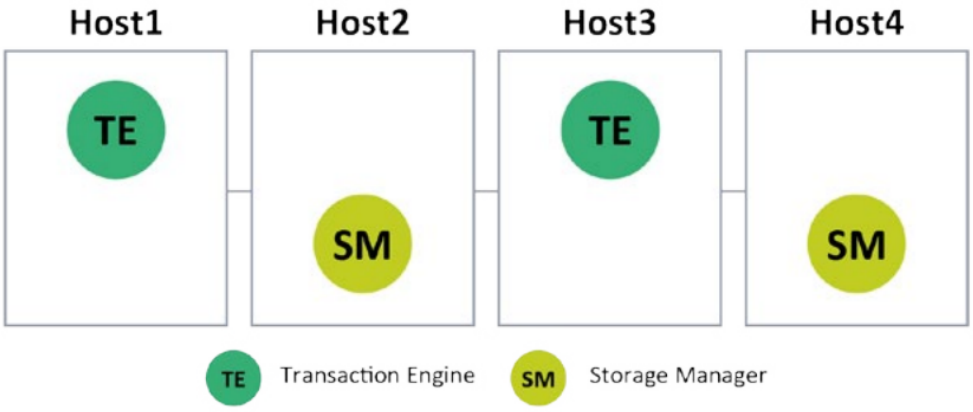

Figure 3: Example of processes running on different host achieving Minimal-Redundancy

On demand scaling is achieved by dynamically creating and shutting down hosts. Migration is also an easy process: Create a new process, wait for it to start that stop the old process. The Database is thus always available as it does not suffer from down times.

[3]Secure Remote Password protocol

3.2.3 Durability

Abstracting all data into Atoms is done in part to simplify the durability model. All access and caching in the architecture is on the Atom-level, and all Atoms have some unique identifier, so Atoms are stored as key-value pairs. By design, durability can be done with any file system that supports a **CRUD** interface and can hold a full archive of the database.

In addition to tracking the canonical database state in its archive, SMs also maintain a journal of all updates. Because NuoDB uses MVCC, the journal is simply an append-only set of diffs, which in practice are quite small. Writing to and replaying from the journal is efficient.

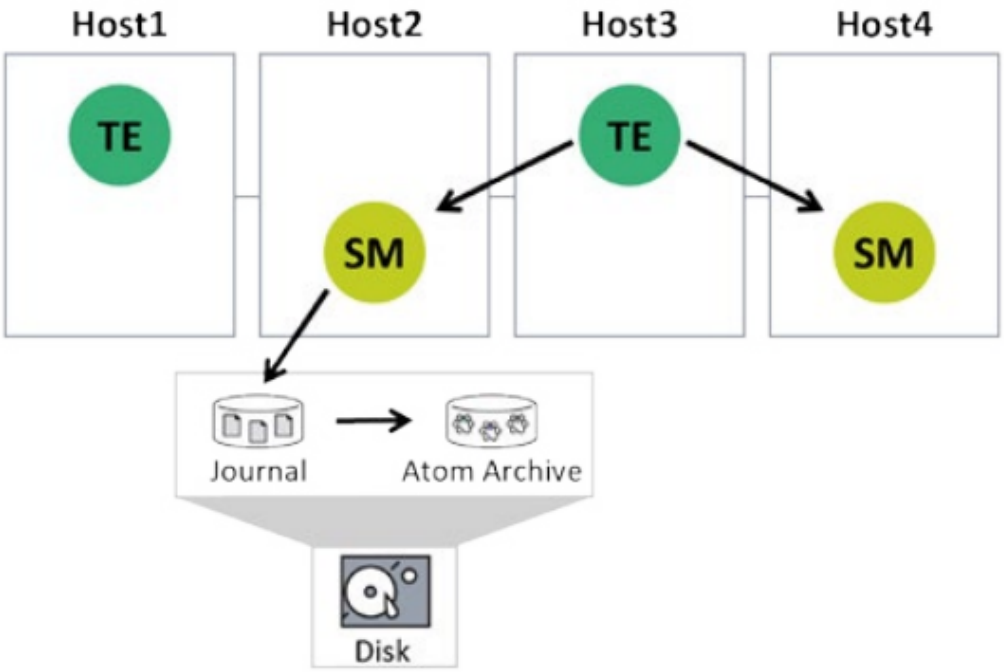

Figure 4: The TE sends transactional changes to all SMs, which records updates to the journal and the archive, while committing the change to disk.

3.2.4 A Tunable Commit Protocol

To acknowledge successful commit to an SQL client, NuoDB must ensure that all properties of an ACID transaction have been met. In NuoDB, users can make a trade-off between durability and performance with fast but transient storage in-memory to slower but more durable on-disk persistent storage. This is referred to as the commit protocol.

All transactional changes are sent to all peers that need to know about the change. As discussed above, that means the changes are sent to any TE with the associated Atoms in-cache and all SMs. The fastest method for committing the changes is when reliable messages have been sent asynchronously to all interested peers. As long as all of the SMs don't fail simultaneously at this moment, then this ensures the data will made durable. Regardless, data will always be correct and consistent. For some applications this kind of k-safety (a measure of fault tolerance) at the transaction tier and eventual durability at the storage tier is sufficient. Many applications, however, want to know that data has been made durable on at least one SM before acknowledging commit to the client. This is tunable by running with a Remote:N setting. In this context, N is the number of SMs that must acknowledge data has been made durable before commit can be considered successful. For instance, Remote:2 requires acknowledgement from at least 2 SMs.

4 Our experiment

4.1 Use-case

In order to benchmark NuoDB and its capabilities pertaining to OLTP transactions, we have decided to mimic the behaviour of the popular social network Twitter. Hence, we created a database which allows us to simulate the data Twitter users would generate and read. The schema can be seen in the following figure :

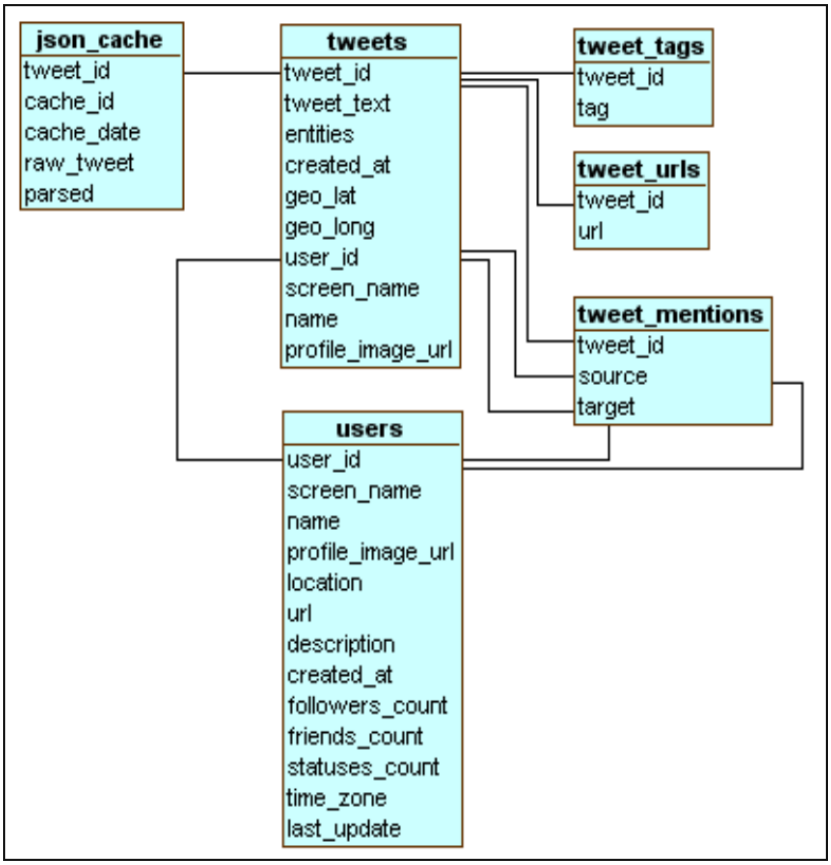

Figure 5: Database schema

The reason why we chose this use-case in order to test NuoDB is due to the fact that we have a typical intensive OLTP application (big number of small transactions) which will put the atomicity to the test. Moreover, the massive throughput of data will also allow to benchmark the elasticity and scalability of NuoDB.

4.2 Environment

The current stable release of NuoDB as of December 18, 2017is 3.0.1.6 . NuoDB offers several versions of it's database. For obvious reasons, we have chosen the **Community Edition** (which is free as in beer). It is important to note that the community edition has several restrictions: A limit of 1 **Storage Manager**, 3 **Transaction Engines** and 500 GB of data.

We have used Amazon's EC2 (Amazon Elastic Compute Cloud) in order to install our Servers in the cloud. We have 3 different server t2.micro instances with **RedHat 7.4**

as the Operating System. The t2.micro have 1GB of RAM, 1 virtual CPU and 15GB of SSD secondary storage. The different hosts communicate via the internal network.

Instance 2	i-074fe986ca52507b7	t2.micro	eu-central-1b	running	Initializing	None	ec2-52-59-188-49.eu-c...	52.59.188.49
Redhat 7.4	i-0c31053876f06ef29	t2.micro	eu-central-1b	running	Initializing	None	ec2-35-159-26-217.eu-...	35.159.26.217
Instance 3	i-0f6e868d709542bb2	t2.micro	eu-central-1b	running	Initializing	None	ec2-35-159-11-229.eu-...	35.159.11.229

Figure 6: The three EC2 instances which compose our database

The storage manager is run on one host and each of the three hosts have a **TE**. A recap of the configuration can be seen in the following figure.

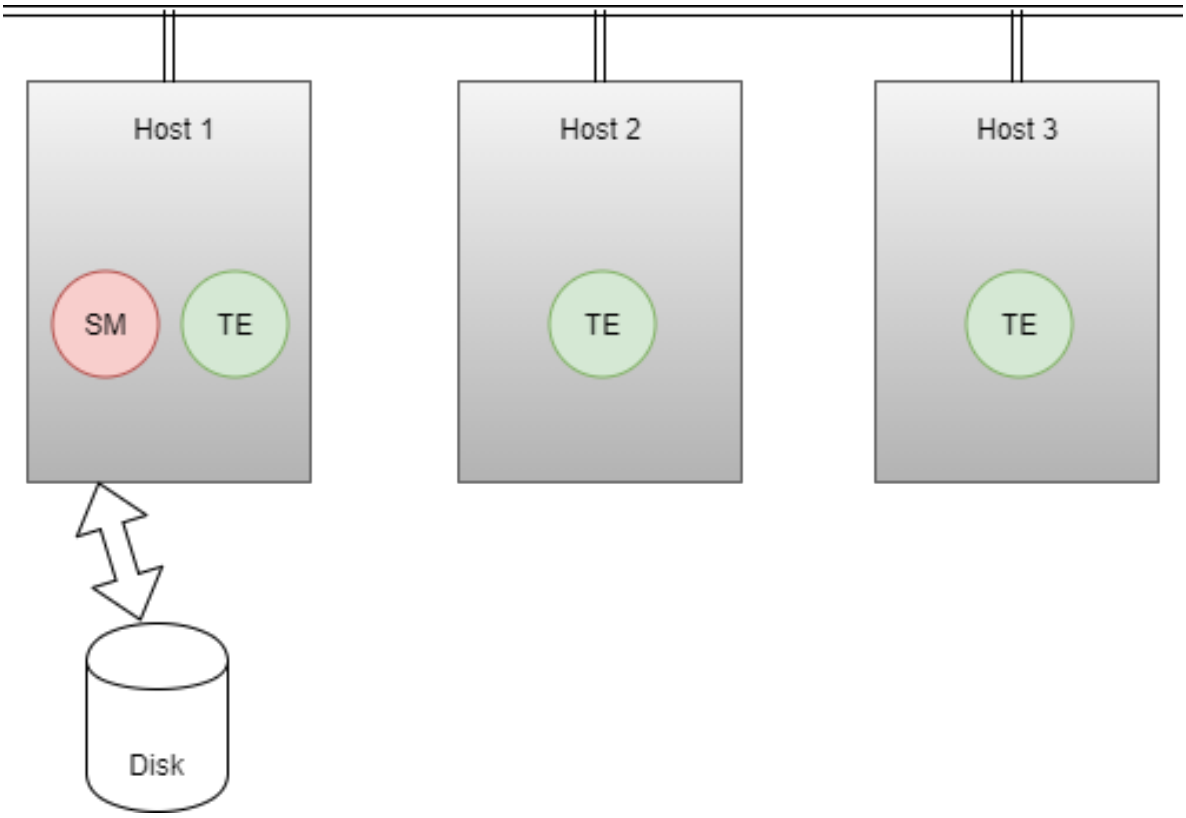

Figure 7: The architecture of our database

In order to generate load that mimics Twitter, we have a simple Java client that inserts tweets in the Database and we measure the time it takes for a number of clients and a number of transactions that are given as input to the program. The clients send requests in parallel thanks to threads.

For the sake of comparison, we have also installed a MySQL server on Host 1 and we run the exact same test for both NuoDB and MySQL.

4.3 Results

We have conducted two tests. We simulate 150 users writing and reading 100 tweets each :
When using NuoDB, the the tweets submission (150 times 100 tweets) takes 4.769 whereas to run whereas MySQL takes 7.797. Reading (150 * 100 tweets) takes 1.83s on NuoDB and 2.10s on MySQL.
Note that these values are an average of 500 runs.

We have also run a test where 200 clients write 100 tweets each.
NuoDB takes 6.43s (write) and 2.437 (read).
The MySQL server crashes because it is out of memory. Because of the fact that our NuoDB server has 3 TE instances, NuoDB has an automatic load balancer. Hence why the NuoDB system does not run out of memory. This shows that scaling our database allows us to sustain good performance and guarantee availability.

5 Conclusion

Traditional database are powerful, clearly the most used databases, but our systems evolve, and our databases need to evolve too. We need more speed to make better real time analysis, more scalability to respond to the new problematic of big data. First we had NoSQL which is faster, can be distributed easier but does not comply to ACID properties which is very a important feature to maintain consistency of our data and is crucial to most applications.

We know have a new alternative, the NewSQL databases, that provide ACID property and can use common SQL language like traditional relational databases. Moreover, it has comparable performance to NoSQL. We have showed that proof that in certain cases (namely OLTP oriented applications), NewSQL is a better solution than traditional relational databases like MySQL.

NewSQL databases are still fairly new and lack broad adoption. However, they are promising and may become the solution for new businesses in the near future.

References

[1] Nuodb technical white paper. http://go.nuodb.com/rs/nuodb/images/Technical-Whitepaper.pdf.