A Comparison of SQL and NoSQL Database Management Systems (DBMS)

1. Introduction

Both SQL type Relational database management systems (RDBMS) and NoSQL DBMS are collaboratively used for production environment. The below content will be focusing on comparisons between those two DBMS in terms of schemas and flexibility, ACID compliancy, scalability, availability, throughput, security and typical applications.

2. Schemas and flexibility

Firstly, when creating the schemas, RDBMS often needs to mention the primary key foreign key constraint for table join SQL. So when there is need to change the scheme later, the table will break. For example, the below CREATE statements illustrates the one to many relationship between the Driver and Vehicle TABLE.

```
CREATE TABLE Driver (
   DriverNumber INTEGER NOT NULL,
   Email          VARCHAR(20) NOT NULL,
   CONSTRAINT driverNumber _pk PRIMARY KEY(driverNumber  ))
CREATE TABLE Vehicle (
   VehicleNumber   INTEGER NOT NULL,
   Status  VARCHAR(10) NOT NULL,
   DriverNumber   INTEGER NOT NULL,
   CONSTRAINT vehicleNumber_pk PRIMARY KEY(VehicleNumber),
   CONSTRAINT driverNumber _fk FOREIGN KEY(DriverNumber )
     REFERENCES Driver (DriverNumber) )
```

Secondly, Cassandra does not have join, so tables are not constrained by primary key foreign key. So changing the driver table will not influence the vehicle table. So it is more flexible than RDBMS . Its schema would be like this :

```
CREATE TABLE driver (
driver_name TEXT,
email TEXT,
PRIMARY KEY (driver_name)) ;
CREATE TABLE vehicle (
vehicle_id TEXT,
status TEXT
PRIMARY KEY (vehicle_id));
```

Thirdly, MongoDB is schemaless, document based. Document is like a JSON structure, it uses referencing to find nested object and more flexible than Cassandra. Field constraints and referential integrity constraints are not required in MongoDB. A database in MongoDB contains different collections which contains list of documents. Each document has similar structure with each other in the same collection but some variation is allowed. For example:

```
{ "_id" : 1, "driver_name" : "a", "status" : 10, "position" : 2 },
{ "_id" : 2, "driver_name" : "b",  "skills" : [ 1,2,3] },
{ "_id" : 3, "driver_name" : "c", "email" : "driver_name@mail.com" }
```

3. ACID compliancy

Firstly, atomicity of a database system means the indivisible property of the operation.

RDBMS requires if any part of the transaction fails, the whole transaction fails and rolls.
Cassandra's write and delete operations are atomic at the partition level. For example, setting a write in QUORUM level and 3 replicas. It will replicate the write to all the three nodes.
Mongodb's atomic write operations are at the document level, which only influences one document in one collection.

Secondly, consistency in database system means a certain database transaction can only change data in an allowed way according to different level of requirements.

RDBMS requires database constraints will not be violated before and after transactions are made.
Cassandra allows user to set eventual or tunable consistency to determine how strong the level can be for each transaction.
Mongodb provides eventually consistency to long time of no writing operations to synchronize the data between the slave node and the master node in the replication set.

Thirdly, isolation is about the degree visibility of transaction integrity to the client.

RDBMS allows concurrent operations against the database without inconsistency.
Cassandra performs isolation at the row level which is only seeable to the client calling that node.
Mongodb uses server side write lock and read lock to implement isolation mechanism.

Finally, durability is referring how well the database can preserver the committed changes.

RDBMS will keep the data permanently after the transactions regardless of the system crash or not.
Cassandra uses commit logs to restore any lost writes if the server crash memtables flushed to disk.
Mongodb by default commits the database files in every one minutes and the journal operations in every 0.1 seconds. These default feature can be configured via syncdelay and journalCommitInterval configuration features.

4. Scalability

All the DBMS can scale up by horizontal and vertical scaling.

Horizontal scaling is by adding more nodes to the distributed cluster to achieve higher computing capability.

According to Sankaran (2015-06) "Distributed Database for Multi Mediation", vertical scaling can be achieved by upgrading the hardware like CPU and memory to a single computer and updating the software like the operation system. However, this approach lacks of resiliency since it is single point failures and it usually involves migrating the database from one cheaper machine to a more expensive machine.

However, RDBMS is harder than NoSQL DBMS to scale due to the constraints of the ACID compliancy. The scaling solution for RDBMS is very expensive.

Cassandra allow that the performance of a service does not rely on the amount of users and client using that service. Some of the vendor support auto scaling and other allow users to scale up by using some external API.

Cassandra's simple design schema allows horizontal scaling easier by adding nodes to the cluster. Due to Cassandra's consistent hashing algorithm, horizontal scaling is a better solution for Cassandra than vertical scaling. When more capacity is required, nodes can be added to Cassandra easily by computing the address of the new nodes in the hashing ring. This kind of flexible scaling allows Cassandra to deploy in both commodity hardware and cloud based platform like DAAS(database as a service), SAAS(software as a service), PAAS(platform as a service), IAAS (infrastructure as a service).

MongoDB provides three approaches for scaling.

First one is cluster scaling, which allow users to distribute their databases into hundreds of nodes hosted in MongoDB providers within multiple data centers. For example, EA sports FIFA computer game is using the cluster scaling to support its gamers all around the world.

The second approach is the performance scaling which increases the read and writes per second with strict service level agreement. For example, Qihoo uses performance scaling to support over 100 applications with 20 billion operations one day.

The third one is the data scale which stores billion documents in the database. For example, Craigslist uses MongoDB's data scaling to handle 80 million of ads posts one month.

5. Availability

Availability means that DBMS will respond clients with non error for every request, which means if a system is in high availability, it has high successful rate of read and write operation during a period of time.

According to the CAP (Consistency, Availability, Partition Tolerance) theorem, a DBMS can not meet those three requirements at the same time. For distributed computing system, partition tolerance is always required. So in most cases, either consistency or availability is chosen for NoSQL DBMS design.

RDBMS with the ACID principle usually prefers consistency than availability, where consistency of the data records in database are extremely important for business like banking transactions.

For NoSQL DBMS like Cassandra and MongoDB, which follow the BASE(Basically Available, Soft state, Eventual consistency) guarantees, choose availability than consistency. For example, giant online shopping platform like Amazon, they would like their users to be able to add items to their shop cart even when the system is down. Their strategy is to email their users later to finalize if they can physically place the order or not due the actual stock in the inventory. In the case, they prefer availability of the system to let user finish the online shopping for better User experience.

Cassandra utilizes the replicas factors to achieve different levels of availability. By setting the consistency level in Cassandra, different levels of availability can be chosen. The most common consistency level are strict consistency with low level availability, strong consistency with middle level availability and eventually consistency high level availability.

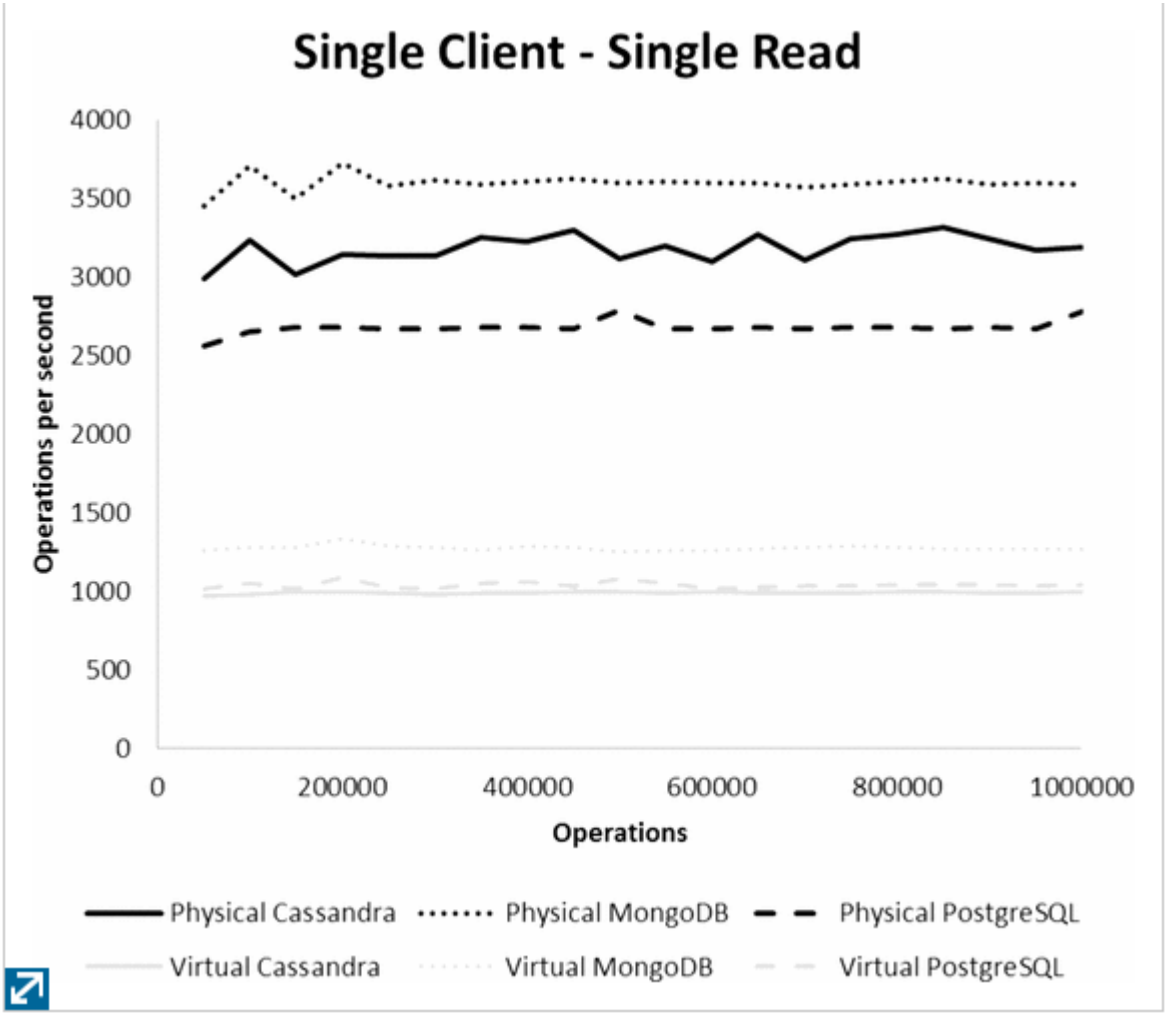
MongoDB utilizes its replica set to provide high availability to clients. They are set of mongod process which keep the same data set. When the primary nodes is not available, the secondary node will become the new primary node to make sure the database is always on.

In terms of the position in the CAP theorem, MongoDB is preferring Consistency and Partition Tolerance than Availability. Cassandra prefers Availability and Partition Tolerance than Consistency. Postgresql prefers Availability & Consistency than Partition.

6. Throughput

Throughput is as known as query throughput in terms of DBMS, which is an indicator to measure the performance of the system. For example, in terms of Online analytical processing (OLTP) type relational database like MySQL , throughput means how many inquiries from customer's request the DBMS can handle per second.

According to Meijer (2012, June) "Sensor data storage performance", Here is the comparison among Cassandra and MongoDB and PostgreSQL.



(From "Figure 6. Single client, single read" of "Sensor data storage performance")

The line chart shows that the throughput of PostgreSQL has the lowest operations per second(around 2500), while Cassandra is in the middle which is around 3000 operations per second, compared with MongoDB which has the highest fluctuating around 3500.

7. Security

Security in DBMS means the methods to protect the system from illegal use and vicious threat.

Common security mechanisms for RDBMS are detection which utilize audit systems, prevention which is via groups & role assignments and recovery which implements system backup periodically.
Take the MS SQL Server for example, its security system manages all the access by the most important components which are authentication and authorization. Authentication is the mechanism to only allow legit DBMS user (usually DBA) to log in the system. While authorization is to let specific group of users to perform certain set of operation with permissions. For example, the most commonly used permission statements are GRANT, REVOKE and DENY.

Cassandra has five main features to manage the security:
1.It use rolename and user passwords to authenticate accessibility of the keyspaces and tables.
2.Roles in Cassandra cluster can have the right to make operation to the database or table level by issuing CQL authorization commands like GRANT and REVOKE.
3.Cassandra manages instances resources of Java Virtual Machine by Java Management Extensions.
4. Secure sockets layer (SSL) is used for clients and clusters communication encryption.
5. Cassandra closes non-essential firewall ports to restrict port access from external resources.

MongoDB provides db.auth() method to authenticate a user. For example:
var credentials = {user: “Adrian”, pwd : “swen432”, mechanism: “SCRAM-SHA-1”, digestPassword: true}
db.auth(credentials) ;
MongoDB implements Role Based Access Control (RBAC) to manage access to its database system.
Besides basic authentication and authorization mechanism, MongoDB provides TLS(Transport layer security) to encrypt its network traffic.
For enterprise level features, which are accessible in MongoDB Enterprise, for example, is the Federal Information Processing standard (FIPS) mode, which will provide users a government standard for data encryption and decryption.

Most of the RDBMS are hosting in the inhouse data center, while the NoSQL DBMS are hosting in the cloud which are distributed data centers all around the world. For example, companies which use Amazon SASS to deploy their database, will have multiple datacenters assigned by Amazon across the world, which means the client companies can not have fully control to their data. Beside, some governments have the right to access of any data stored in cloud, which means some business sensitive data stored on cloud NoSQL database may be compromised.

8. Typical applications

RDBMS is used for both OLTP (Online Transaction Processing) and OLAP (Online Analytical Processing), in which the former one is emphasising on ACID properties, which is reliable for transactional business activities like banking and manufacturing. In contrast, the later one OLAP is good at behavioural business data analysis, for example , big data industry.

PostgreSQL is an application example of using RDBMS as a data warehouse solution for OLAP. The general features of PostgreSQL are connectivity which empowers PostgreSQL for the data mining integration and extensibility which allows users to define functions like stored procedures and customized domain and data types. The scalability/updatability of PostgreSQL can be implemented by PL/Proxy which is partitioning system implemented as procedural language. Consistency of PostgreSQL can be enforced by Serializable transactions and explicit blocking locks. PostgreSQL security is implemented via user authentication ,user names and group , access control , functions & rules and secure TCP/IP connection. PostgreSQL servers work seamlessly to let a secondary DB server to take over rapidly when the primary one is down to achieve high availability. It can also allow multiple computers to store the same data to achieve load balancing. Both network latency and numbers of clients influences the throughput. According to 2ndQuadrant (2011) when the latency is 0.07 ms in 1Gbps type network, the transactions/sec reaches 14286 and it is only 3333 per second in 0.3 ms latency 100Mbps network.

Hadoop is a framework for big data solution, which consists of Hadoop Distributed File System (HDFS) , MapReduce API and open source data analysing project (HBase, PIG). Its scalability can be achieved by the distributed processing spanned on clusters of nodes. Under the hood, the divide and conquer algorithm is used to break the data ceiling. The consistency strategy for is “one copy update semantic”. Object stores in HDFS are typically eventually consistent. Operations like CREATE, DELETE and UPDATE take time to let the caller informed thus there is no grantee to inform the clients instantly the changed just made. Hadoop provide secure mode to secure the platform which it is off by default. Data confidentiality, service level authorization, authentication, authentication for web consoles are the security features to for hadoop to secure the server. Running more than one redundant NameNodes in the same cluster allow to HDFS to achieve high availability. There will always be exactly one of NameNodes is in an active state while the others will be in the standby state, in which the former one is accountable for all caller’s request in the cluster, while the standby nodes are acting as slaves to keep adequate state to support a failover. For Hadoop, number of records queued to be processed are so large that the latency is irrelevant so optimizing via low throughput is better for performance.

9. Conclusion

Even RDBMS and NoSQL have differences in terms of those seven factors, they can work together very well. In the long term future, those two type of database will be used interchangeably at the same time. Industry still needs to rely RDBMS for transitional operation while NoSQL for distributed computing like big data. So NoSQL is not a replacement for RDBMS, instead, it is a complement.

References

NSR Sankaran Kuruganti (2015-06) “Distributed Database for Multi Mediation”. Retrieved from http://www.diva-portal.org/smash/get/diva2:825934/FULLTEXT02

Van der Veen, J. S., Van der Waaij, B., & Meijer, R. J. (2012, June). Sensor data storage performance: SQL or NoSQL, physical or virtual. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 431-438). IEEE.

2ndQuadrant (2011-11-11) “PostgreSQL Durability & Performance”. Retrieved from https://wiki.postgresql.org/images/3/3b/2011-11-11_PostgreSQL_SyncRepPerformance.pdf