Author: Dylan Clement

Scylla DB analysis and comparaison with MySQL / MariaDB

For a long time we have been interested in NoSQL databases in order to improve performances, and have a flexible data model and scalability.
MongoDB was our first choice (towards the end of 2016 – at the beginning of our adventure), but migration hadn't been made due to huge changes to do, our lack of free time at this period, no further studies about it and we have instead put a MySQL replication.
Recently, we are looking again at a migration to a NoSQL database, for the same reasons and because it is better suited for organization features that we are developing.

Cassandra seems good for us with its ability to handle large amounts of data across many servers, its performance compared to other NoSQL databases and its high availability with no single point of failure. CQL is close to SQL, it will be easier for us editing the code for migration.

An important point: users must have a performance gain in uploading and downloading files.

We will use DataStax PHP Driver, implemented like an extension and which uses C/C++ driver.
This choice is motivated by the fact that DataStax is a main contributor of Cassandra project and their drivers are maintened.

Not long after, we discovered Scylla which is compatible and similar to Cassandra but written in C++ instead of Java. It offers significantly higher throughputs and lower latencies and we will verify this in our tests.

Data consistency

MySQL

https://stackoverflow.com/questions/25690976/does-mysql-replication-have-immediate-data-consistency

“Replication in MySQL is asynchronous by default, or "semi-synchronous" at best. Certainly it does lag in either case. In fact, the replication slave is always lagging behind at least a fraction of a second, because the master doesn't write changes to its binary log until the transaction commits, then the slave has to download the binary log and relay the event.

But the changes are still atomic. You can't read data that is partially changed. You either read committed changes, in which case all constraints are satisfied, or else the changes haven't been committed yet, in which case you see the state of data from before the transaction began.
So you might temporarily read *old* data in a replication system that lags, but you won't read *inconsistent* data.
Whereas in an "eventually consistent" system, you might read data that is partially updated, where the one account has been debited but the second account has not yet been credited. So you *can*see inconsistent data.”

Author: Dylan Clement

Cassandra

There is in the docs a section about data consistency on Cassandra: https://docs.datastax.com/en/cassandra/3.0/cassandra/dml/dmlDataConsistencyTOC.html

To sum up, consistency on Cassandra is tunable: the client may decide the appropriate consistency level (zero, any, one, two, three, quorum, all, etc.). The consistency level controls both read and write behavior based on your replication factor. In a single node cluster the consistency levels any, one, quorum and all are equivalent.

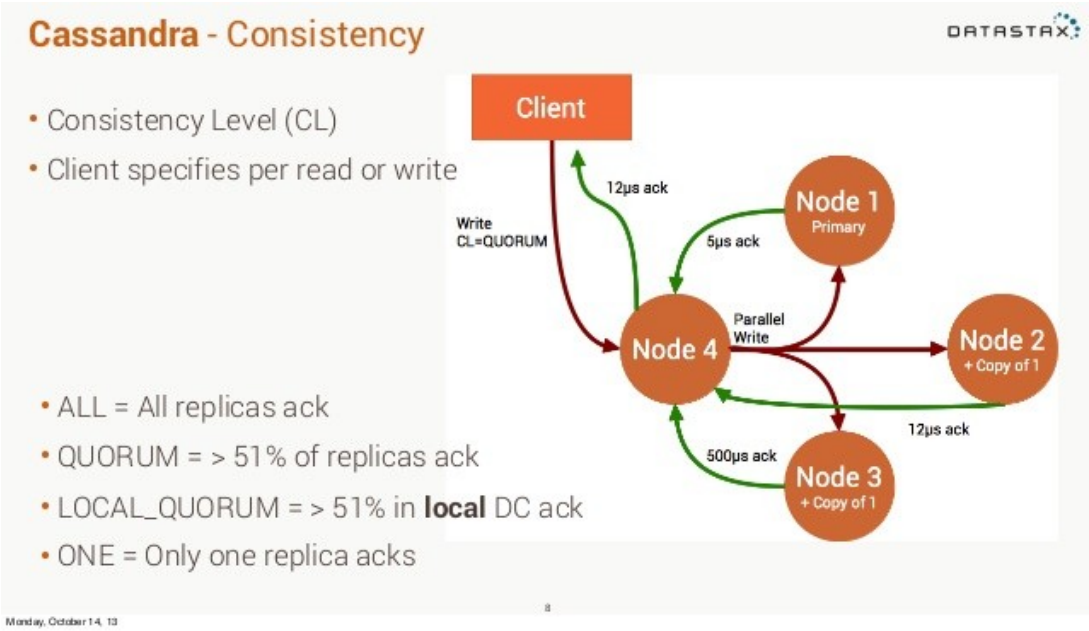


Figure 1 – Cassandra consistency - jeremiahdjordan - https://www.slideshare.net/jeremiahdjordan/cassandra-intro-27171544

Tests

We set up VMs with Ubuntu 16.04, 4 GB of RAM, 16 GB storage on HDD, 1 CPU core per machine (i5 4690K). PHP 7.0 and Apache2, Last stable version of MariaDB / Cassandra / Scylla according to the machine. Caches are disabled.

Tests consist in writing 10K rows, reading them one at a time, reading them with a prepared statement, updating them and deleting them. We could have done it with more rows but it already shows the differences significantly.
Cassandra and Scylla are on a single node.
We did several measures for the same operation in order to check the reliability of our results.

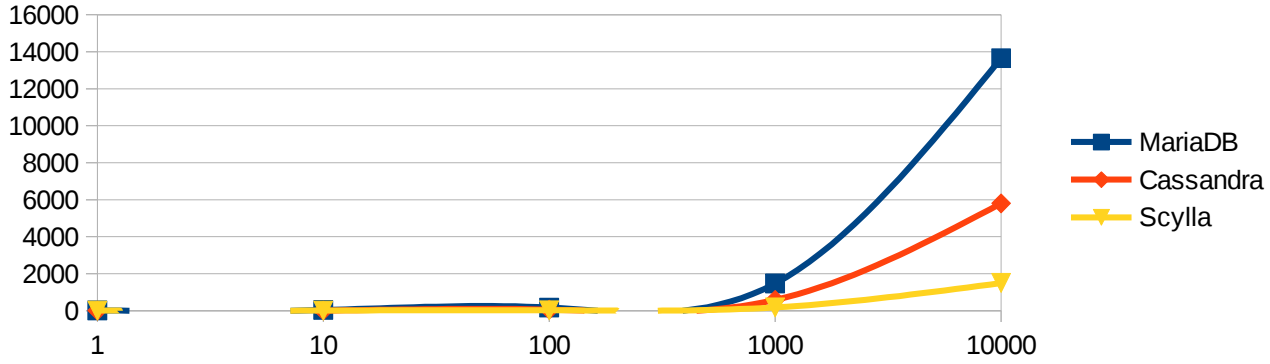
We used the same table (song) with song_id as primary key (int in MariaDB, uuid in Cassandra/Scylla), artist (text), title (text), album (text) and populated with realistic data.

Author: Dylan Clement

Measurements are in ms.

Inserting

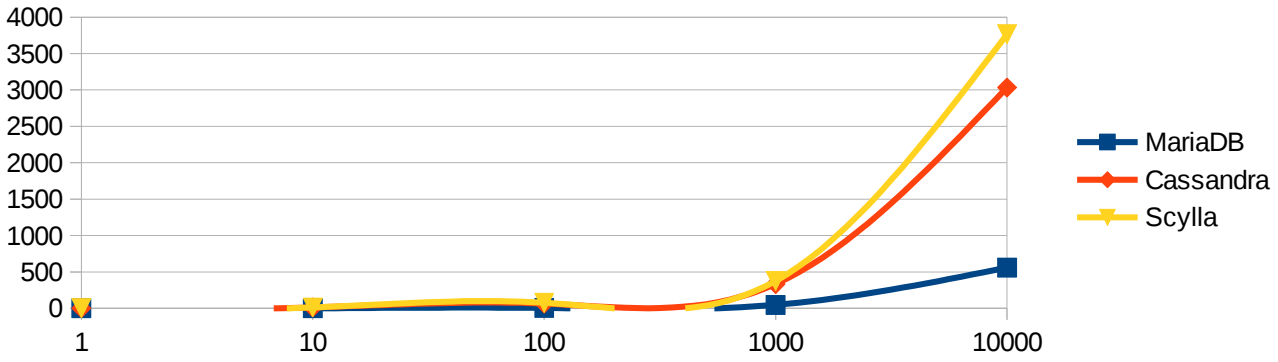
	1	*10*	*100*	*1000*	*10000*	*CPU*	*RAM*	*Operations/s*
MariaDB	17,17	34,67	165,1	1466,92	13651,2	27 %	32 %	733
Cassandra	0,74	4,3	55,73	578,43	5798,53	26 %	31 %	1725
Scylla	1,2	3,76	20,26	174,7	1493,15	26 %	31 %	6697



Scylla is the fastest (3.88 times more than Cassandra and 9.13 times more than MariaDB). Results correspond to our expectations and forecasts.

Reading (one per one)

	1	*10*	*100*	*1000*	*10000*	*CPU*	*RAM*	*Operations/s*
MariaDB	0,06	0,49	4,61	47	558,92	27 %	32 %	17891
Cassandra	0,73	11,9	64,88	333,13	3033,48	26 %	31 %	3297
Scylla	0,78	11,13	76,82	374,7	3765,63	26 %	31 %	2656



Author: Dylan Clement

For reading MariaDB is faster and really good at that. We expected less difference but it should be noted that data got to be modeled differently: with Cassandra/Scylla, we must optimize reading operations by reducing and simplifying them (this example is a simple query) at the expense of writing. This is not a problem if the data is multiplied, as writing does not cost much. For a more complex query, the difference will not be as big and Cassandra could be faster. We can also use caching.

Reading (prepared statement, simple query, 10K rows)

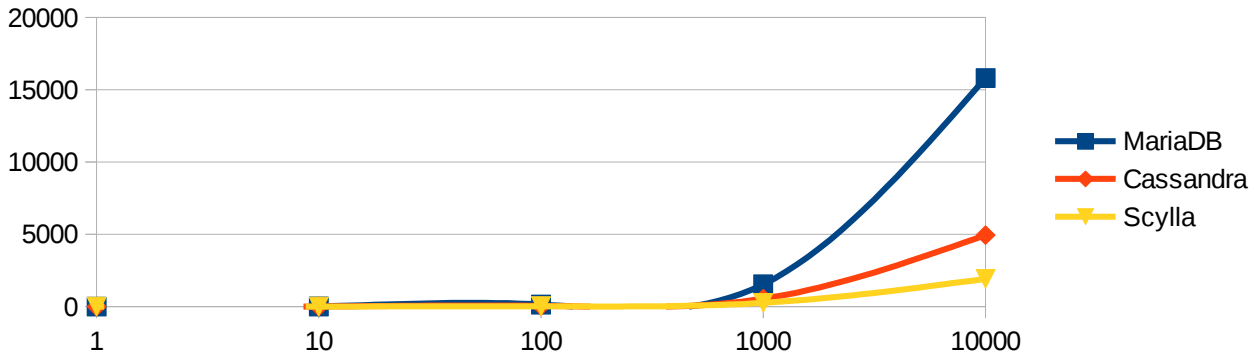
		CPU	*RAM*
MariaDB	4,58	28 %	32 %
Cassandra	174,68	28 %	32 %
Scylla	160,25	28 %	32 %

Reading (prepared statement, query with a JOIN on MySQL)

		CPU	*RAM*
MariaDB	651,13	32 %	34 %
Cassandra	224,3	28 %	32 %
Scylla	208,25	28 %	33 %

Updating

	1	*10*	*100*	*1000*	*10000*	*CPU*	*RAM*	*Operations/s*
MariaDB	2,77	14,02	143,84	1543,6	15811	32 %	32 %	632
Cassandra	1,33	6,51	56,42	551,02	4946,93	32 %	32 %	2021
Scylla	0,51	1,79	23,85	257,92	1918,92	32 %	32 %	5211

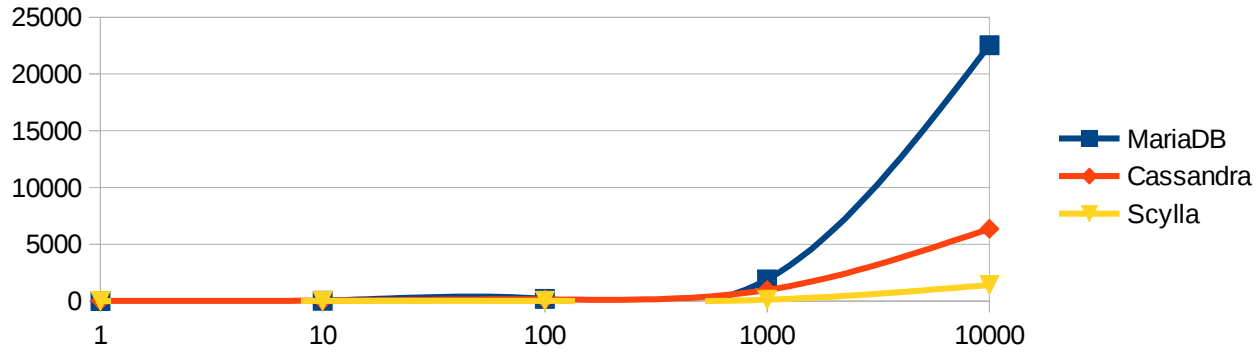


Scylla is the fastest (2.58 times more than Cassandra and 8.25 times more than MariaDB).

Deleting

Author: Dylan Clement

	1	*10*	*100*	*1000*	*10000*	*CPU*	*RAM*	*Operations/s*
MariaDB	7,85	23,45	189,25	1905,78	22529,6	36 %	32 %	444
Cassandra	28,1	39,84	138,25	977,18	6358,24	31 %	32 %	1573
Scylla	0,66	2,97	13,84	122,2	1406,78	31 %	32 %	7108



Scylla is the fastest (4.52 times more than Cassandra and 16 times more than MariaDB).

Conclusion

Scylla and Cassandra are much faster than MariaDB/MySQL except for reading and Scylla is faster than Cassandra.

RAM and CPU usage does not really change according to our tests, deleting is a little more heavier on MariaDB.

For queries, we currently have almost the same number of reads and writes. But most of time, reads are simple queries like getting columns for a given ID.
As mentioned above, with Cassandra/Scylla, we must optimize reading operations by reducing and simplifying them at the expense of writing. Writing does not cost much.

We will have better performances for files/folders management (copy/move/create…).

On the infra-scripts (cron) side, performances are not very important as it does not impact the user. The most important is saving time on uploading/downloading phase.

File upload
incrementSizeStored method is called at every chunk, methods called at file creation or file completed are not significant. So, we will get better performances (write operation – UPDATE).

File download
getUploadFolderPath method is called, then, the file is read on the disk. Only the path is from the database but we use a cache with Redis, so it will not impact performances.

We need to pay attention to different read operations, especially about getting a folder because they are often read and refreshed.

Author: Dylan Clement

Scylla also allows us to have a better scalability and a high availability with no single point of failure.

Scylla data modeling

We had to structure our data in order to keep best performances, so, it is different compared to relational databases. New features such as comments, contacts, share with… have been added too.

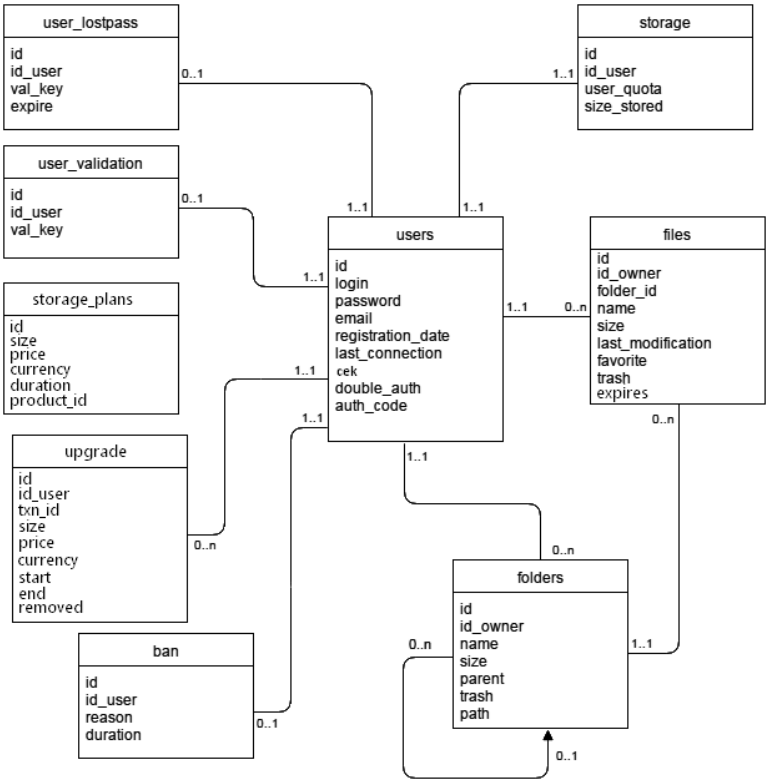


Figure 2 – MariaDB data modeling used at Muonium before migration.

Primary Keys
http://intellidzine.blogspot.fr/2014/01/cassandra-data-modelling-primary-keys.html

This article is very interesting about primary keys. The concept is different in Cassandra/Scylla and we had to think about its purpose. So, we will talk below about compound primary key, partition key, cluster key.

Author: Dylan Clement

Modeling

Foremost, we could put user_lostpass, user_validation and storage into users as they involve users and a single row per user. But we decided to keep them separated because we do not use them at the same time and they have a different application. They have a simple primary key (user_id).

The data type of IDs will be “uuid” on Scylla and “timestamp” for dates. Tinyint(1) columns will be stored as “boolean” data type.

For users table, we added “pks” which is used to store public and private asymmetric keys.

Concerning file; owner_login (used at file sharing) and path are stored instead doing joins to get them.
“lek” and “dk_asym” are keys used for file comments and internally shared files.
“updated_at” is a clustering key, used for the recent files feature.
We added “updated_at” for folders too, although it is not a clustering key.
contacts contains a list of users for a given user.
shared_with contains a list of users for a given file.
comments contains an ID, file, user, added_date and its content, with comment_id as partition key and added_date, file_id as cluster keys.

Author: Dylan Clement

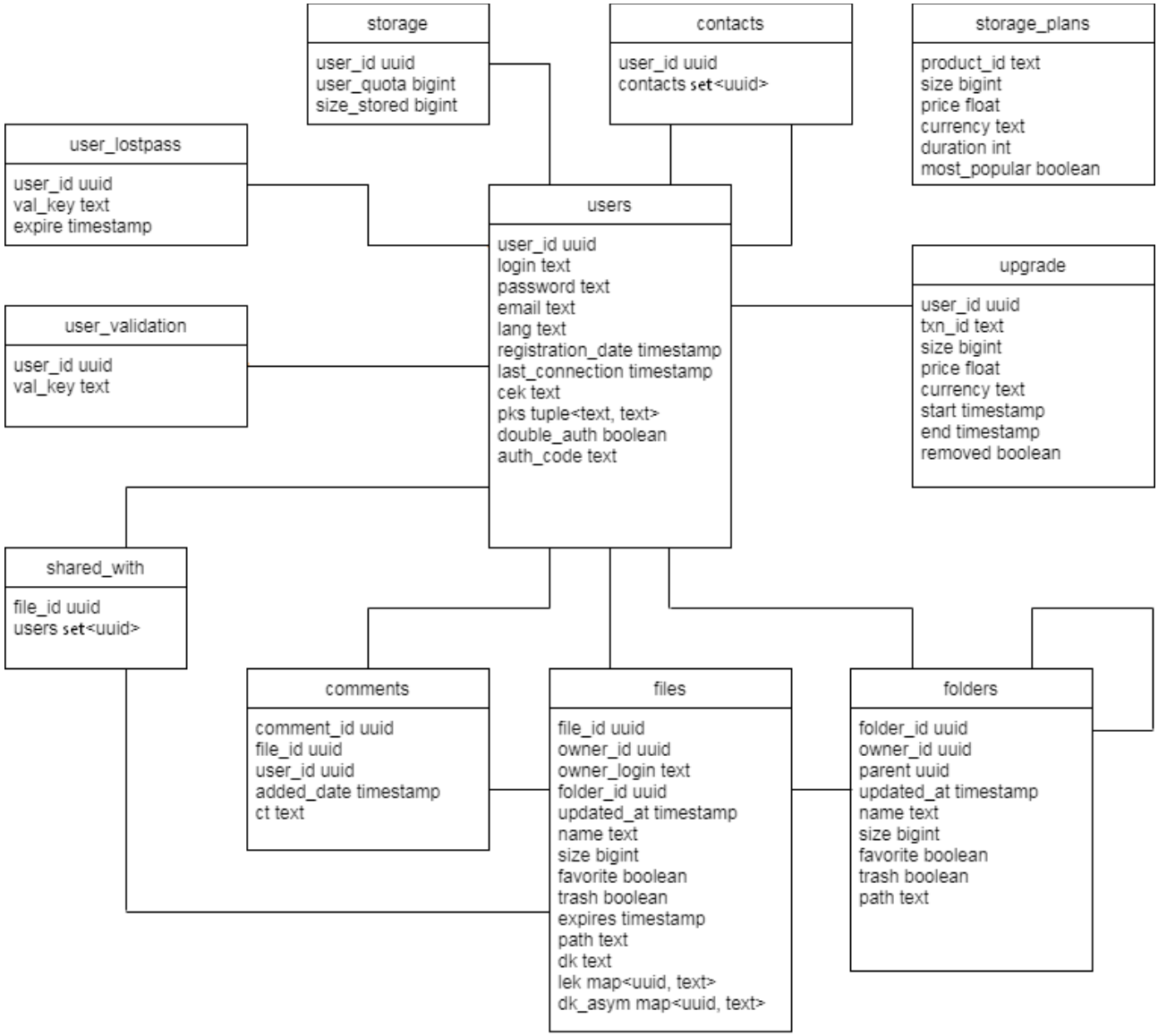


Figure 3 – Scylla data modeling