Handling Big Data: Performance Comparison for MySQL, MongoDB and Redis

Anastasiia Karpenko
University of Trento

Göksel Tolan
University of Trento

Queries

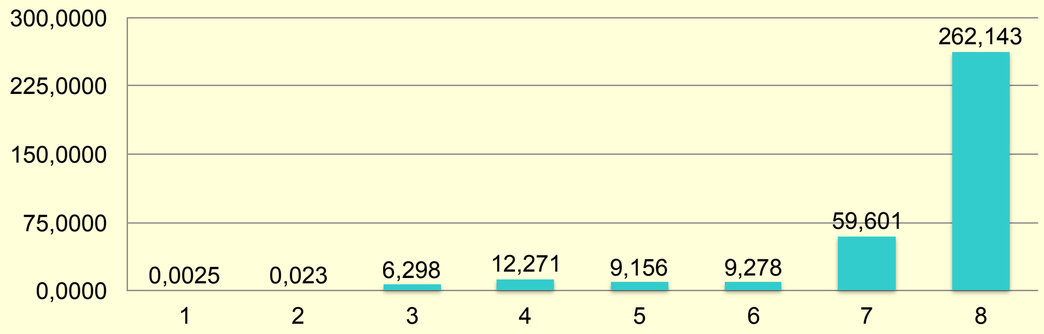
As for the operations, 8 various queries have been chosen, according to the *Yahoo! Cloud Serving Benchmark (YCSB) framework*, that represent basic operations that can be performed with data of 7 million rows, which is 689.4MB

Query	MySQL (SQL) queries	Type of query
1. Print all the information stored in table	SELECT * FROM ontime;	Standard, read workload
2. Print destinations stored in the table	SELECT dest FROM ontime	Standard,read workload
3. Print the number of flights for each of the Tail Number	SELECT TailNum, COUNT(FlightNum) as num FROM ontime GROUP BY TailNum;	Aggregate, read+modify workload
4.Update the weather delay column for the flights which were delayed on more than 30 mins.	UPDATE ontime SET WeatherDelay = 10 WHERE DepDelay > 30;	Modify workload
5. To calculate various information	SELECT Month, sum(AirTime) as sum_AirTime, avg(DepDelay) as avg_depdelay, avg(AirTime) as avg_airtime, max(AirTime) as max_airtime, min(AirTime) as min_airtime, count(*) as count_order FROM ontime GROUP BY(Month);	read/modify workload
6. To calculate various information for each month ordered by average departure delay time.	SELECT Month, sum(AirTime) as sum_AirTime, avg(DepDelay) as avg_depdelay, avg(AirTime) as avg_airtime, max(AirTime) as max_airtime, min(AirTime) as min_airtime, count(*) as count_order FROM ontime GROUP BY(Month) ORDER BY avg_depdelay;	read/modify workload
7. Add new column to the dataset table	ALTER TABLE ontime ADD AirTimeHours double;	Insert workload
8. Transform minutes into hours and add to the newly added column	UPDATE ontime SET AirTimeHours = AirTime / 60 WHERE 1 = 1;	Mostly modify workload

Machine Configuration

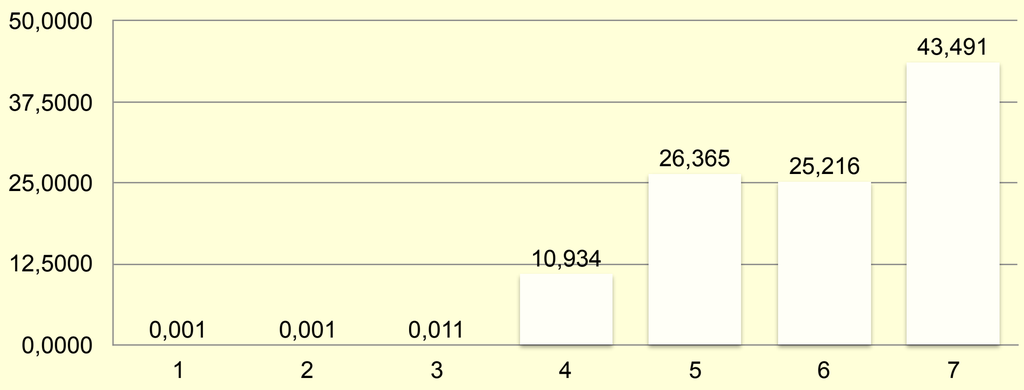
For this project implantation MacBook Pro 13 was used with the following configuration:
-2.7GHz Dual-core Intel Core i548GB 1866MHz LPDDR3 SDRAM
-256 GB PCIe-based Flash Storage
-OS X El Capitan

MySQL



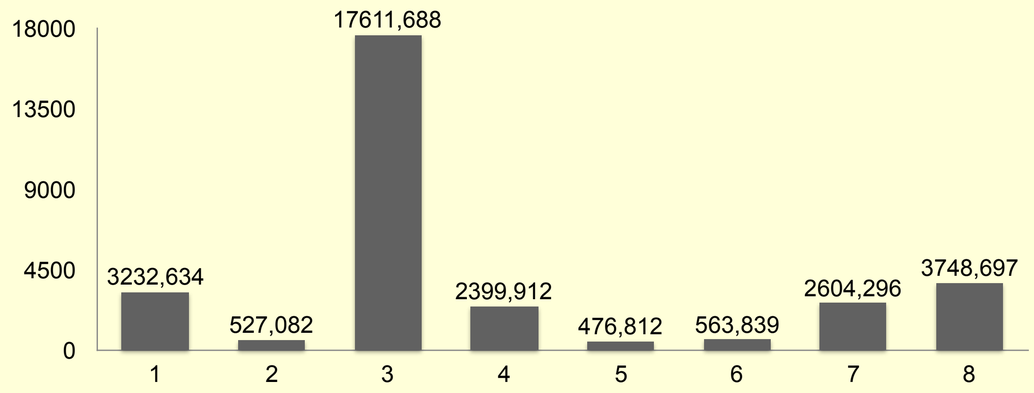
As it is shown in the graph above, MySQL works slower with modify-write type of queries.(number 7 and 8). All the results are shown in seconds.

mongoDB



According to our results, it is clear in the graph that MongoDB performs slower with modify-write type of queries.(number 7). Even if MongoDB is better with reading, its slower than MySQL while dealing with aggregation operations. All the results are shown in seconds.

redis

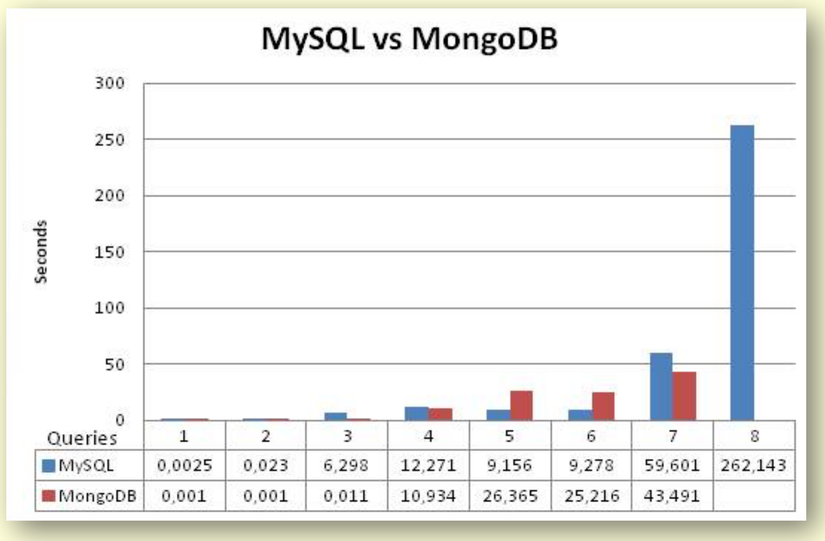


It is obvious that according to results that we have taken, there is huge time difference between Redis and the others; MongoDB and MySQL. And Redis is much slower than the other 2 databases. All the results are shown in seconds.

Experiments & Results

Queries	MySQL, seconds	MongoDB, seconds	Redis, seconds
1	0,0025	0,001	3232,634
2	0,023	0,001	527,082
3	6,298	0,011	17611,688
4	12,271	10,934	2399,912
5	9,156	26,365	476,812
6	9,278	25,216	563,839
7	59,601	43,491	2604,296
8	262,143		3748,697

According to our results since there is a huge difference between Redis and the others; MongoDB and MySQL we decided to compare MongoDB and MySQL together with another graph below.



If we compare performance of MySQL and MongoDB that are more or less the same (except Redis), we will see that for 2 of the queries (5,6) which are quite similar and do a lot of mathematical aggregation operations MySQL is much more efficient than MongoDB. For the other majority of queries MongoDB is more efficient and fast. We can notice that for read type of queries (1,2) MongoDB shows better performance. Also queries 3 (read+aggregate+write) ,4 (modify +write) and 7 (modify+write) are faster in MongoDB.

Conclusions

In order to improve the research it could be interesting to implement this project with different dataset sizes from smaller to bigger. Also using more sophisticated queries will may give interesting results.

Acknowledgement

Our thanks to professor Yannis Velegrakis for motivation and encouragement. Thanks to our colleagues Tiziano Antico, Andrea Galloni, Ana Daniel and Daniel Bruzual for support.