Sentiment analysis and time series prediction on Twitter data

Alex Santini

MSc in Data Analytics

Integrated assignment big data processing and Advanced data analytics

CCT

28.10.2023

Sentiment analysis and time series prediction on Twitter data

Introduction

*SQL and NoSQL databases comparison*

*When it comes to storing SQL and NoSQL databases are the two main categories a developer needs to know for choosing the right tool for the right job; especially in the case of managing text data the two architectures have their strengths and weaknesses, In order to test those a testing strategies have been developed through the use of a familiar benchmarking tool called YCSB on a Linux virtual machine.*

*This comparison aims to explore the performances of a SQL database (Mysql) and a Nosql database (MongoDB) when taking a read-heavy and a insert-heavy load.*

*The databases in questions will have to contain the same tabular format to equally compare them, the format chosen is the same present in the excel sheet provided for the rest of this analysis in the csv file (projet\_tweets.csv) with the schema as follow:*

|  |  |  |
| --- | --- | --- |
| **Field** | **type** | **Example** |
| id | INT | 4587 |
| date | DATE | Sat May 16 23:58:44 UTC 2009 |
| flag | STRING | NO\_QUERY |
| user | STRING | bobthebuilder |
| text | STRING | Lyx is cool |

*Fig.1 YCSB comparison schema*

It’s important to notice how the schema must be explicitly mentioned prior to loading the data in the MySQL database whilethe same is not a requirement for MongoDB showing the first noticeable difference between the two.

The testing strategy aim to compare a classic insert heavy load and a more generalist *read heavy load with the following parameters:*

|  |  |
| --- | --- |
| Read Heavy |  |
| Count | 10,000 |
| Read | 80% |
| Update | 5% |
| Insert | 1% |
| Read modify write % | 2% |
| Scan | 3% |
| Distribution | Zipfan |

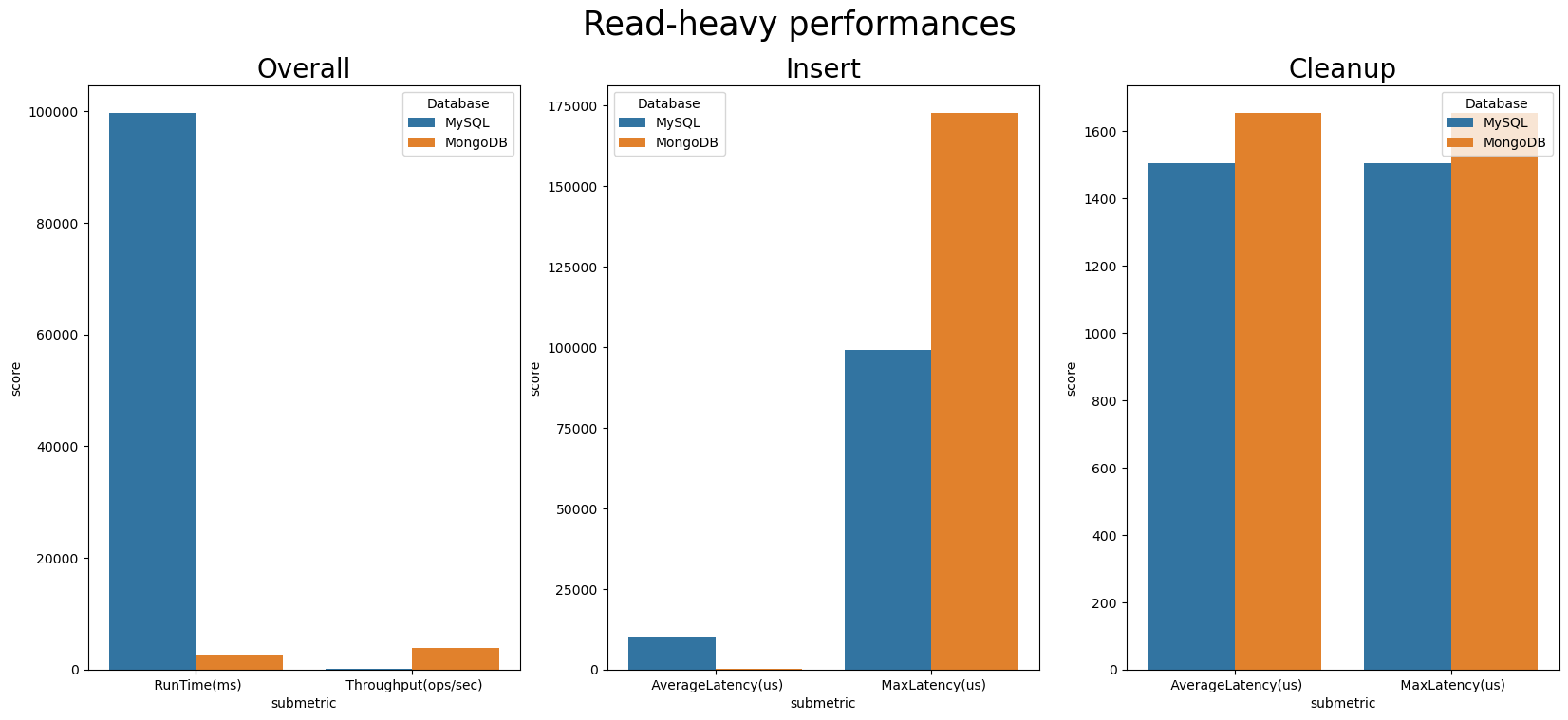
*Fig.2 Read heavy custom load parameters.*

In order to test the scalability of the two architectures a 80-20 insert-heavy load has been tested for different sizes as shown in Fig.3.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Count | 5,000 | 10,000 | 50,000 |
| Insert | 80% | 80% | 80% |
| Read | 20% | 20% | 20% |

*Fig.3 Insert heavy loads.*

By providing both read and insert comparison the test strategy aims to achieve a good comparison of SQL and NoSQL functions in a text heavy context.

As expected the results shown a clear difference between the performances of the two databases; the main metrics took in consideration for this approach were the overall runtime measured in milliseconds and throughput of the database measured in operations per second: this two metrics were chosen because considered good overall indicators of overall performances when it comes to choosing a database for a production pipeline, other metrics like maximum latency and average latency where taken also in consideration but as secondary point od reference*Fig.4 Read heavy results*

The result for the read heavy workload are clear as MongoDB shows better performances running the load 20 times faster than the SQL counterpart, this is highly expected as being a document-based database MongoDB can handle large volume of read operations outputting a higher throughput than MySQL as shown in figure; MySQL however seems to perform slightly better in cleanup operations and in minimizing the maximum latency in insert jobs which in this case were only counting as 1% of the total workload.

Second step of the testing strategy was to compare three different workloads by increasing the number of total operations (5k,10k,50k) in order to test the scalability of an insert heavy architectures in SQL and NoSQL models.

The result once again proven how MongoDB is more robust in scalability especially in terms of runtime: MySQL seems to increate its runtime almost exponentially comparing a 5k workload with a 50k workload while MongoDB increasing just slightly its runtime, the same is true for the total throughput of the databases where MongoDB managed to increase it as needed for larger workloads while MySQL showing a performance plateau in fig.5; it’s also important to notice how even is increasing MongoDB performances seems to be affected by diminishing returns in throughput signaling how scalability might become an issue for NoSQL databases after a higher threshold.

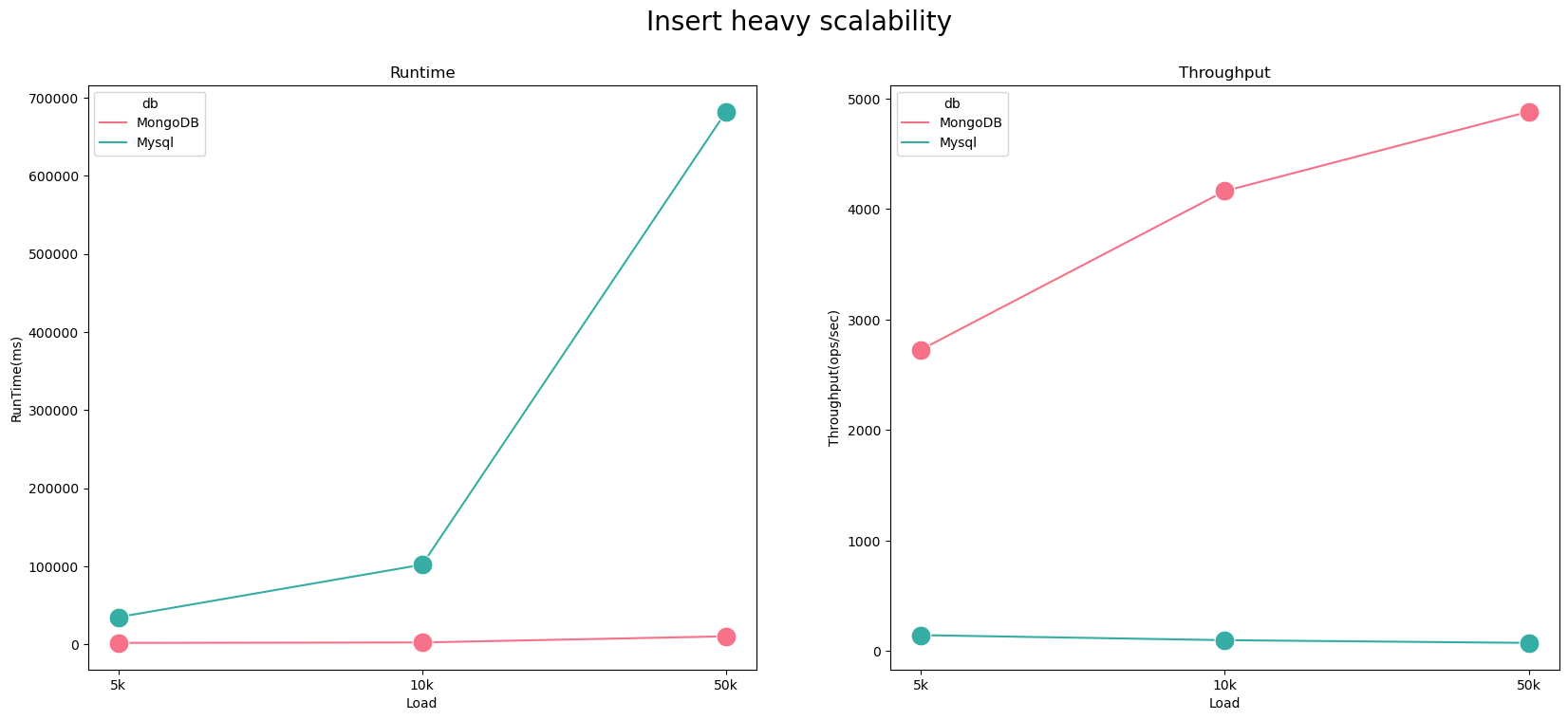


Fig.5 Insert heavy scalability load

*Processing architecture*

*Sentiment analysis*

*Time series prediction*

Reference List

*Seven steps for effective leadership development* (2012) Available at: http://www.oracle.com/us/media1/steps-effective-leadership-dev-1657106.pdf (Accessed: 24 November 2018).

Guenther, R. and Vittori, G. (2012) *Sustainable healthcare architecture*. New York: John Wiley & Sons.

Smith, V. and Jones, R. (2012) ‘Individual assignments and academic dishonesty: exploring the conundrum’, *The Educational Researcher*, 35(1), pp. 37–56.