Analysis of Netflix Database Tests

Overview

This document serves as the report and analysis for the benchmarking of three different Database Management Systems (DBMS). Initially, I did three passes of each CRUD function for records of 1, 1000, 100,000, and 1,000,000 records. However, the performance of the NoSQL queries seemed to bottleneck with the larger records, some even timing out before completion and performing well below expected standards. Because of this, I reassessed my NoSQL CRUD functions.

Upon reevaluation, I found that making all functions asynchronous significantly improved performance due to enhanced concurrency and optimised resource usage. The substantial difference was evident, with the time taken to update 10,000 records reduced from 30 minutes to 0.1 minutes. As a result, I restarted the tests, conducting three passes of each database type again but this time with records of 1, 100, 1,000, 10,000, 100,000, 500,000 and 1,000,000 records.

These extra benchmark points allowed for a better overview for identifying patterns or trends in the data and gauging where any bottlenecks may occur.

The specifications of the test bench were as follows:
AMD Ryzen 7800X3D 4.20 Ghz
32GB DDR5 RAM
MSI Geforce ETX 4080 12GB
Windows 11 Pro 64 bit
6 TB SSD

To ensure data accuracy, I took the median time in milliseconds for each function from all three passes. This created consistency by averaging the durations and eliminating variables such as concurrent processes or system temperatures. I also maintained uniformity by using the same data and entities for each test across all DBMSs.

To create a consistent testing environment, I tried to minimise external factors by keeping other processes running on the PC to a minimum. I also made sure that the same set of processes were active during every test run, thus avoiding any disruption to the execution threads.

Trends

Upon examining the data, there are clear trends.There is a discernible exponential increase in duration when going above 1000 records. In the lower range, there appears to be an optimal performance range where performing an operation on one record, but fewer than 100,000 records, was much faster than performing the operations on one record. This leads me to believe that performing operations on multiple records at once, up to a certain threshold, may be more efficient. Determining this threshold could be another assignment.

Create Operation

The performance of all three database types was relatively comparable initially (figure 2). However, a noticeable change happened when reaching 10,000 records (figure 1). From this point, NoSQL was consistently the fastest. ADO experienced surprisingly large slowdowns when handling 500,000 and 1,000,000 records.

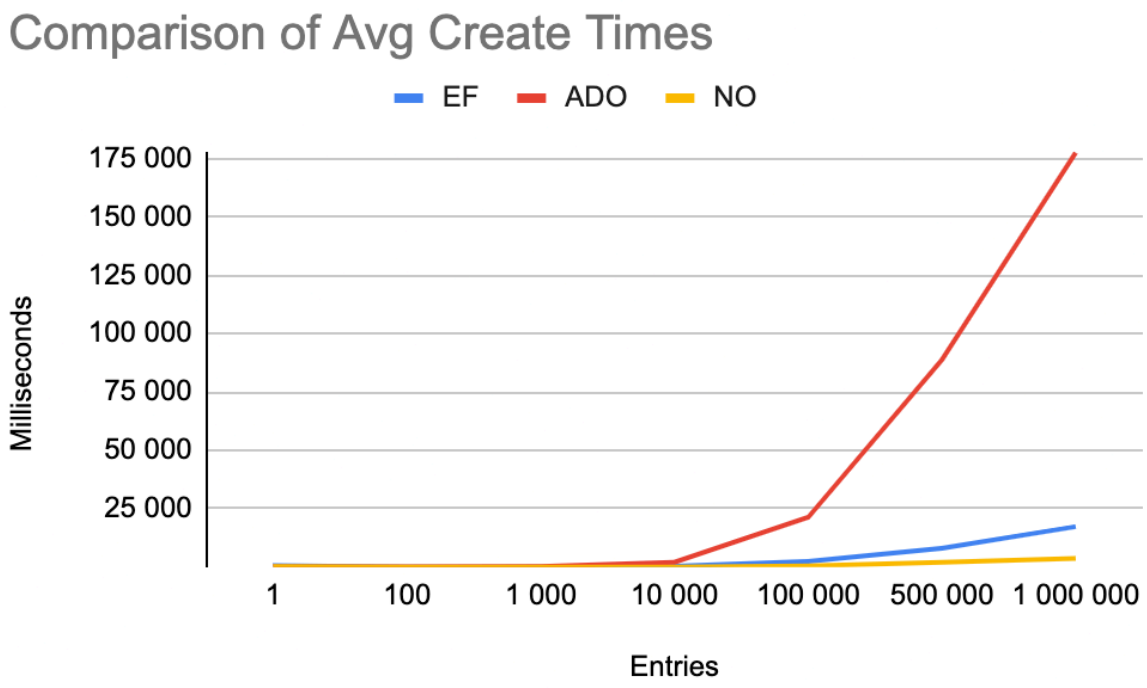

Figure 1

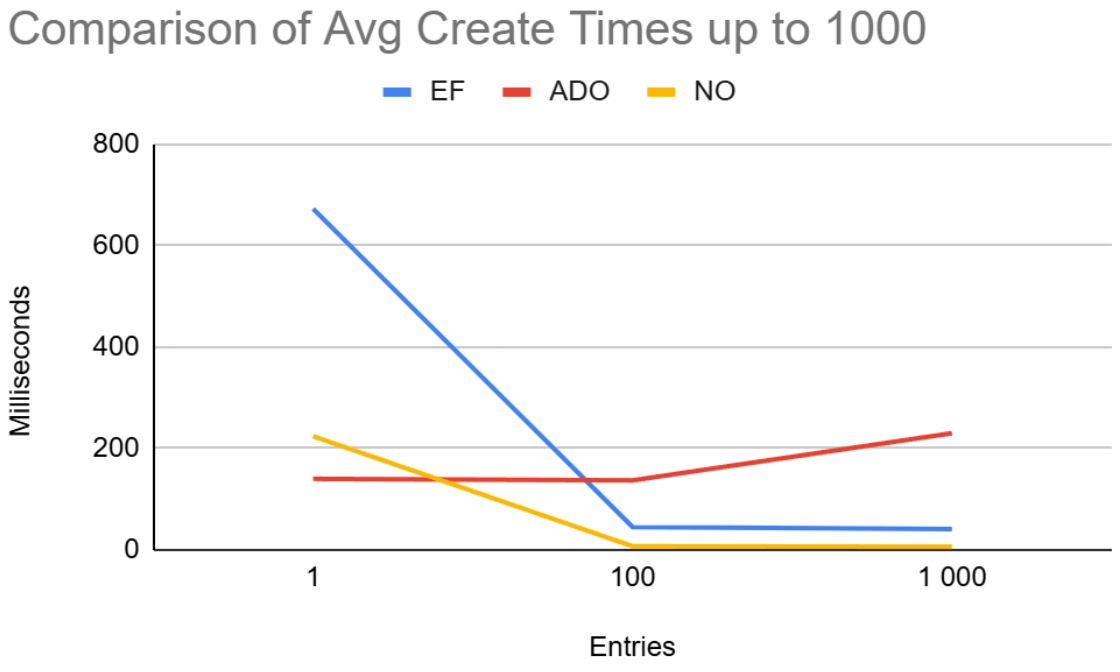

Figure 2

Read Operation

The data shows that NoSQL was the most consistent overall (figure 3). Entity Framework (EF from hereon) performed erratically and was the poorest of all three, especially after 100,000 records where it slowed down significantly (figure 3). ADO was relatively consistent until the 10,000 mark (figure 4), after which it became erratic and slow(figure 3).

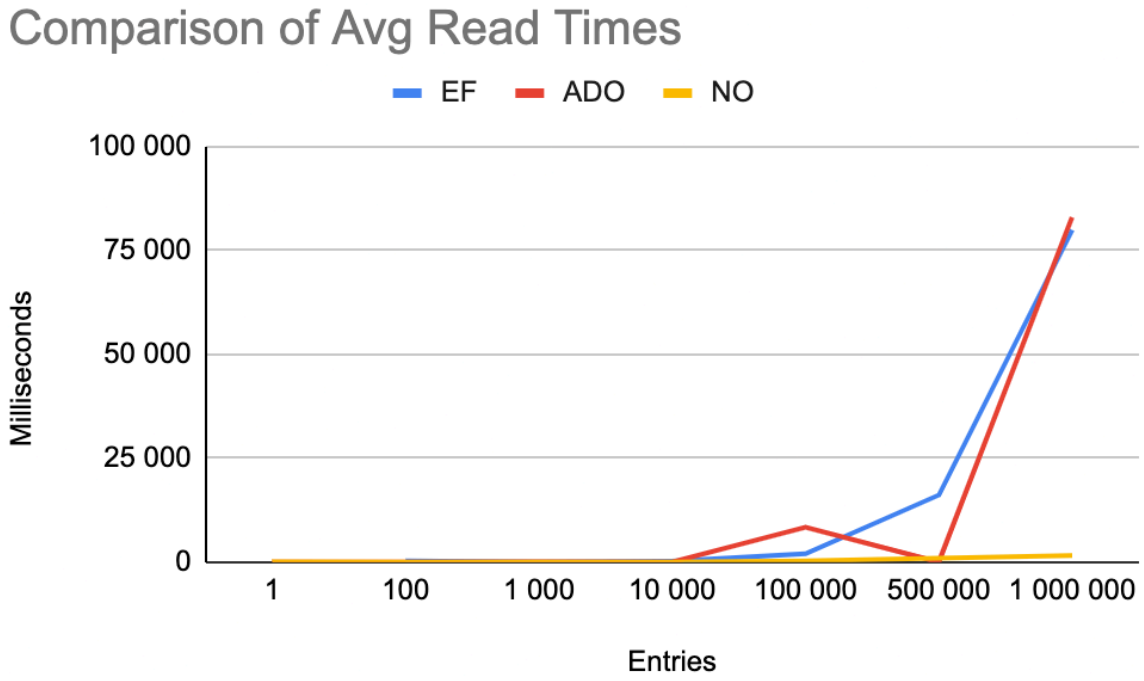

Figure 3

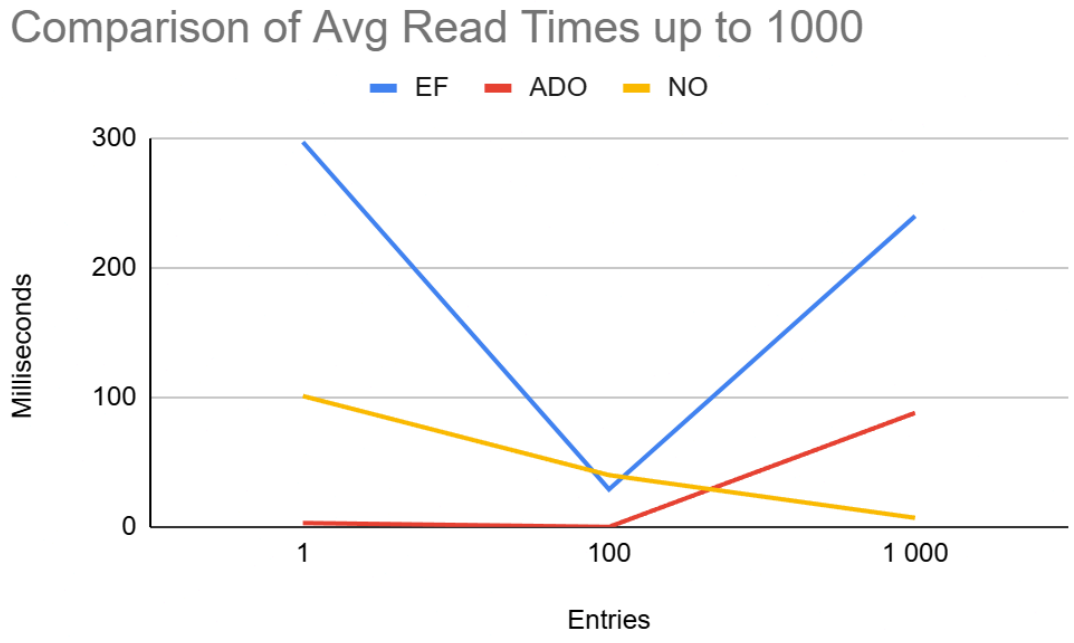

Figure 4

Update Operation

The data for the update operation showed an anomaly, NoSQL slowed down considerably when it reached 10 000 records (figure 5), as opposed to its consistently good performance in the other operations. In this case, ADO was the most consistent throughout the number of records (figure 5).

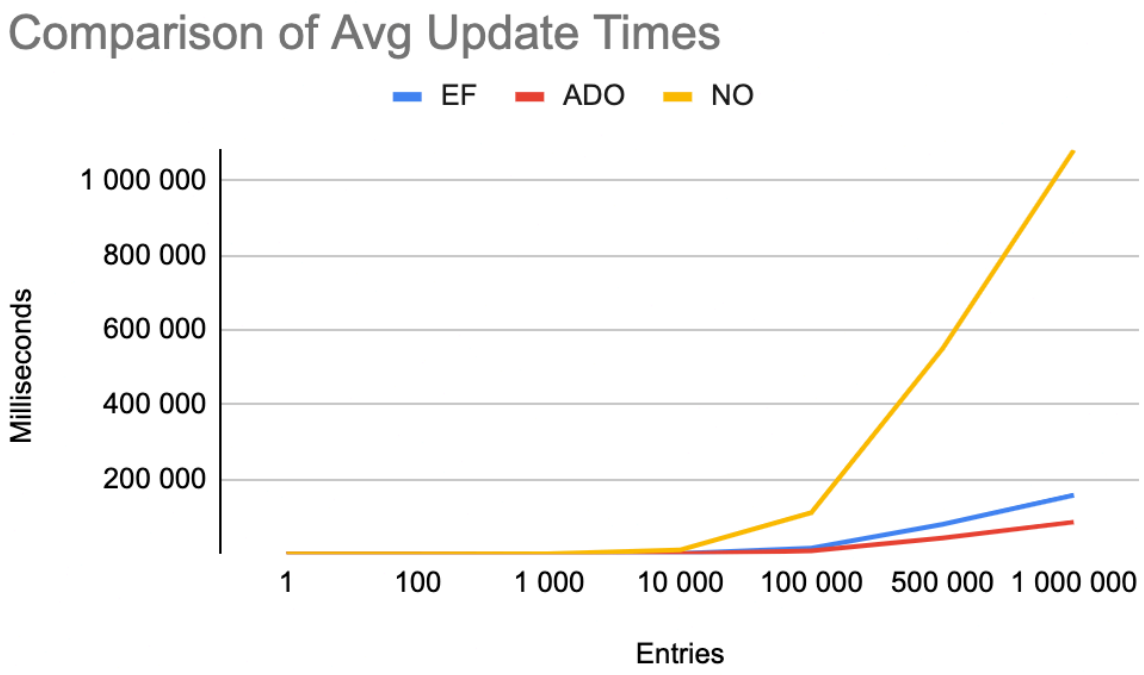

Figure 5

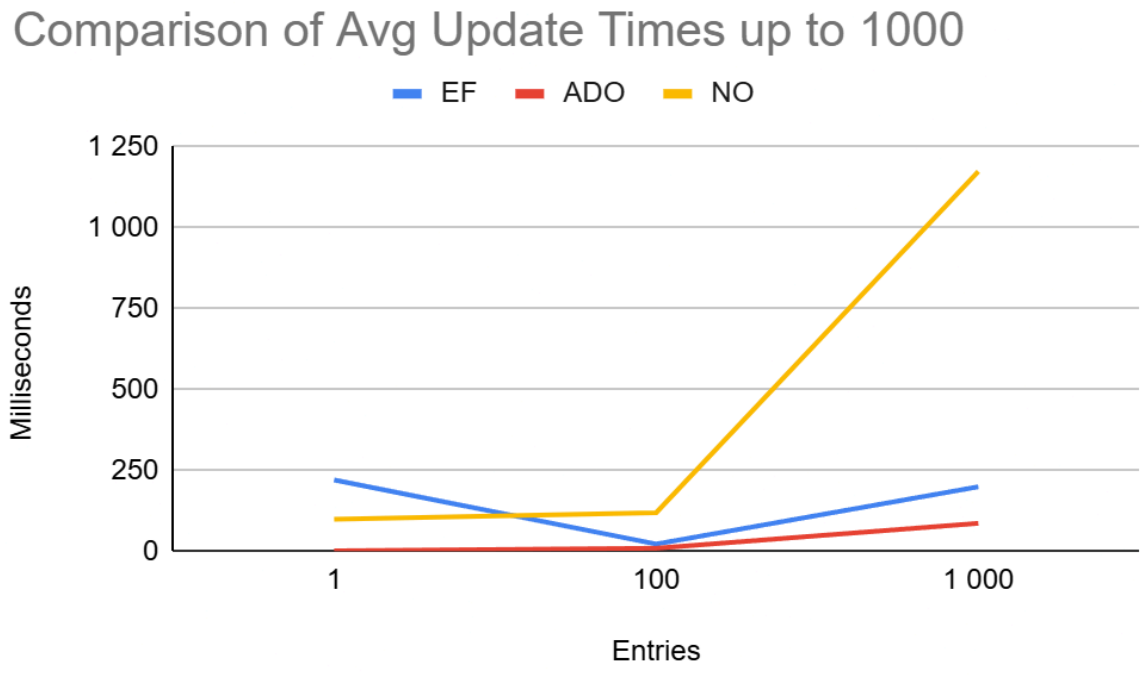

Figure 6

Delete Operation

The data showed that NoSQL was the best performer by a wide margin, with an average time of one millisecond for one million records. EF was erratic throughout the test (figure 8) and ADO was more consistent as the number of records increased and performed comparably to NoSQL(figure 7).

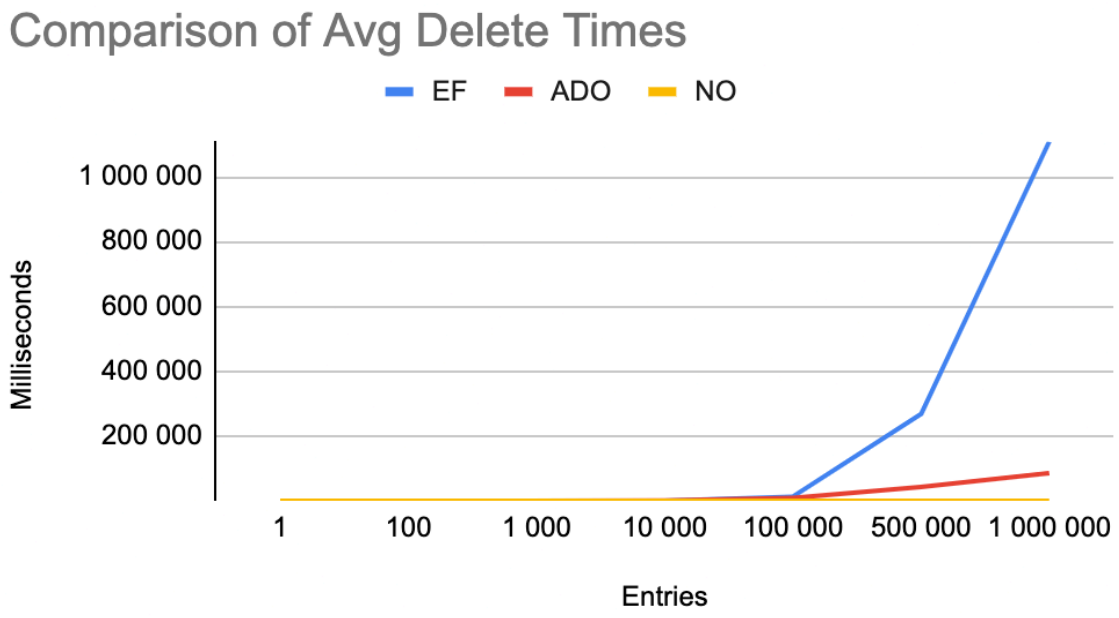

Figure 7

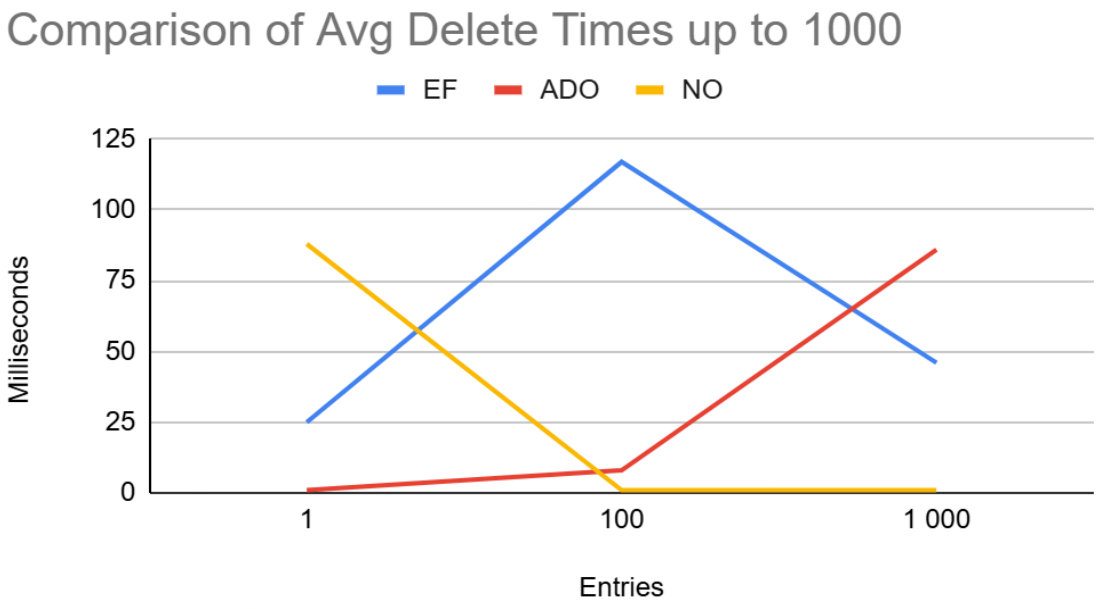

Figure 8

Conclusion

The investigation showed that the three database types had similar performance overall. However, NoSQL consistently outperformed the others in all operations except for the update operation. Entity Framework displayed erratic performance with no clear trend, while ADO demonstrated a more expected performance curve.

Based on results of this investigation, it is recommended that Netflix use a NoSQL database, as it displays faster speeds and is better suited for scalability compared to relational databases. Additionally, its non-relational nature provides greater flexibility for future customisation.

In terms of performance ranking, NoSQL takes the lead, followed by ADO, with Entity Framework ranking last.