The Effect of Software Packages and Platforms on Serialization and Deserialization Times

Ryan Current

Rochester Institute of Technology

STAT 325: Design of Experiments

Linlin Chen, Ph.D.

April 29, 2023

Abstract

The present study aimed to investigate the effects of Package and Platform on Serialization and Deserialization times, as well as their interaction effect. A factorial design was used to analyze the data collected from different platforms, including Console, Server, and WebAssembly (WASM), and different packages, including MemoryPack, MessagePack, Newtonsoft, and System.Text.Json. The analysis of variance (ANOVA) results indicated significant main effects for both Package and Platform on the Serialization and Deserialization times. Moreover, the interaction effect of Package and Platform was also found to be significant for deserialization, but not serialization. This indicates that the effect of package on serialization and deserialization times varied depending on the platform used for only deserialization. MemoryPack package showed the fastest serialization and deserialization times compared to other packages, and this effect was more pronounced on the Console and Server platforms compared to the WASM platform. These findings can have practical implications for developers who want to optimize the performance of their applications by selecting the appropriate Package and Platform.

Introduction

Serialization and deserialization are essential tasks in modern computer programming, allowing data to be transmitted and stored. Serialization is the process of converting data structures or objects into a stream of bytes, allowing them to be transmitted over a network or stored on a file system. Conversely, deserialization is the process of converting the stream of bytes back into the original data structure or object. These tasks are crucial for a wide range of applications, including web services, message passing, and distributed computing.

Serialization and deserialization times can have a significant impact on the performance of an application. Slow serialization and deserialization can lead to increased network traffic, higher CPU utilization, and longer processing times, reducing the overall performance of an application. A poorly optimized program will waste valuable time, negatively impacting customers, and businesses. Therefore, it is essential for programmers to understand how different programming packages and platforms affect serialization and deserialization times, allowing them to optimize the performance of their applications.

In this study, we investigate the effects of different programming packages and platforms on serialization and deserialization times, as well as the interaction effect between these two factors. By analyzing these factors, we aim to provide programmers with a better understanding of how to optimize the performance of their applications, leading to more efficient and effective software development practices.

Data Collection

The data for this study was collected by serializing and deserializing a list of 100,000 sample objects (each contained a string, integer, and boolean value) using all possible combinations of platforms and packages. Each combination of package and platform was tested 4 times, for a total of 48 observations. We collected both serialization and deserialization times, which will be studied separately. The order in which the data was collected was randomized using Microsoft Excel's randomization function to prevent any bias in the order of data collection.

After the randomized order was generated, the serialization and deserialization tasks were executed one by one, and the data was recorded. The time taken to complete each task was measured using the System.Diagnostics.Stopwatch C# class and recorded in a spreadsheet. The organized raw data can be found in figures 1-4.

Table 1

Observations of serialization time

Platform	MemoryPack	MessagePack	Newtonsoft	System.Text.Json
Console	0.036	0.091	0.273	0.101
	0.025	0.020	0.102	0.083
	0.015	0.015	0.124	0.056
	0.017	0.021	0.097	0.049
Server	0.042	0.105	0.225	0.100
	0.036	0.023	0.105	0.089
	0.018	0.020	0.089	0.053
	0.032	0.008	0.087	0.109
WASM	0.324	0.597	2.492	2.061
	0.322	0.391	2.417	2.050
	0.336	0.402	2.229	2.174
	0.351	0.418	2.275	2.163

Table 2

Observations of deserialization time

Platform	MemoryPack	MessagePack	Newtonsoft	System.Text.Json
Console	0.035	0.081	0.234	0.129
	0.029	0.043	0.126	0.116
	0.039	0.070	0.108	0.080
	0.049	0.073	0.107	0.095
Server	0.029	0.051	0.206	0.092
	0.029	0.022	0.109	0.064
	0.014	0.028	0.101	0.090
	0.011	0.014	0.148	0.095
WASM	0.340	0.588	3.971	2.629
	0.320	0.504	3.764	2.746
	0.288	0.521	3.714	2.491
	0.287	0.575	3.549	2.716

Table 3

Mean of observed serialization time

Platform	MemoryPack	MessagePack	Newtonsoft	System.Text.Json
Console	0.02325	0.03675	0.14900	0.07225
Server	0.03200	0.03900	0.12650	0.08775
WASM	0.33325	0.45200	2.35325	2.11200

Table 4

Mean of observed deserialization time

Platform	MemoryPack	MessagePack	Newtonsoft	System.Text.Json
Console	0.03800	0.06675	0.14375	0.10500
Server	0.02075	0.02875	0.14100	0.08525
WASM	0.30875	0.54700	3.74950	2.64550

Method of Analysis

The data collected for this study was analyzed using a two-factorial design, where the two factors were Platform and Package. The analysis of the data was conducted using a two-way ANOVA (Analysis of Variance). Our factors were package (τ), platform (β), and their interaction effects ($\tau\beta$). The model for serialization and deserialization responses is shown below.

$$y_{ijk} = \mu + \tau_i + \beta_j + \tau\beta_{ij} + \varepsilon_{ijk} \begin{cases} i = 1, 2, \dots, a \\ j = 1, 2, \dots, b \\ k = 1, 2, \dots, n \end{cases} \tag{1}$$

μ - grand mean
τ_i - ith level of package factor
β_j - jth level of platform factor
$\tau\beta_{ij}$- interaction effect
ε_{ijk} - random error, $\sim N(0, \sigma^2)$

Our model also has the following parameter constraints.

$$\sum_i \tau_i = 0 \tag{2}$$

$$\sum_j \beta_j = 0 \tag{3}$$

$$\sum_i \tau\beta_{ij} = 0 \tag{4}$$

$$\sum_j \tau\beta_{ij} = 0 \tag{5}$$

We must also ensure that the assumptions of an ANOVA are met for our analysis to be valid. We are looking for our data to be normal, independent, and have roughly equal variances. Each observation was collected independently, and in random order. In addition, our large sample size of 48 observations means that we can assume our sample is normally distributed by the central limit theorem. We will use Levene's test for equal variance to ensure that our data has roughly equal variances with $\alpha = 0.01$. The results are in figures 1 and 2.

Figure 1

Equal variance test for serialization speed

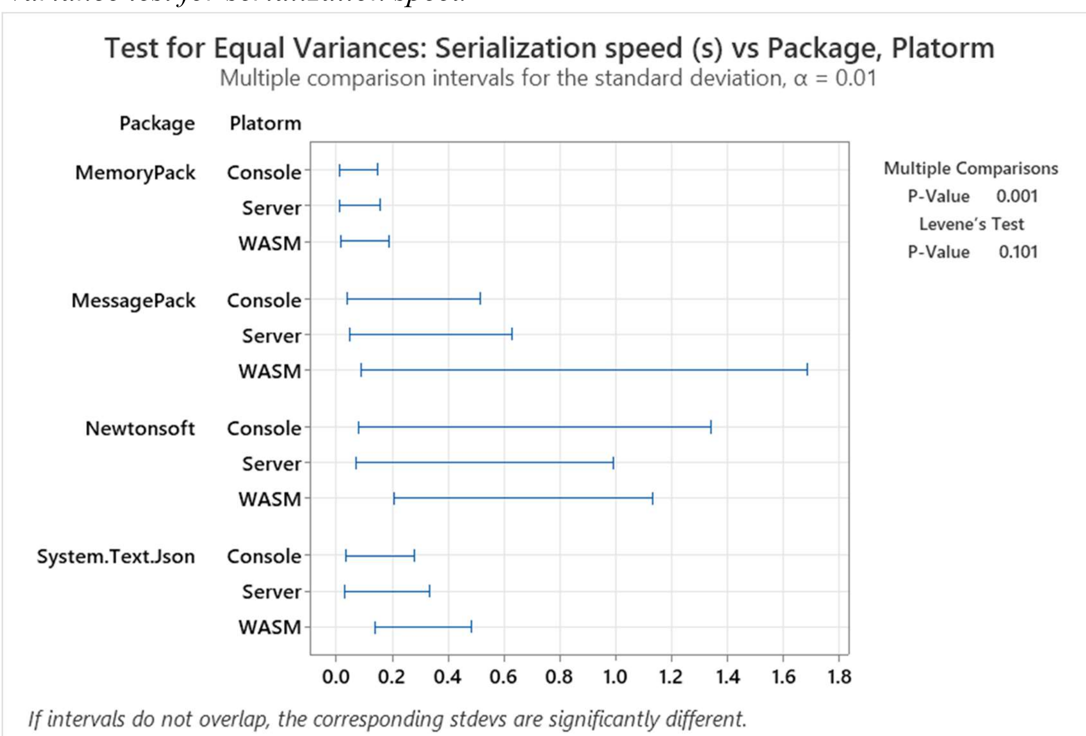
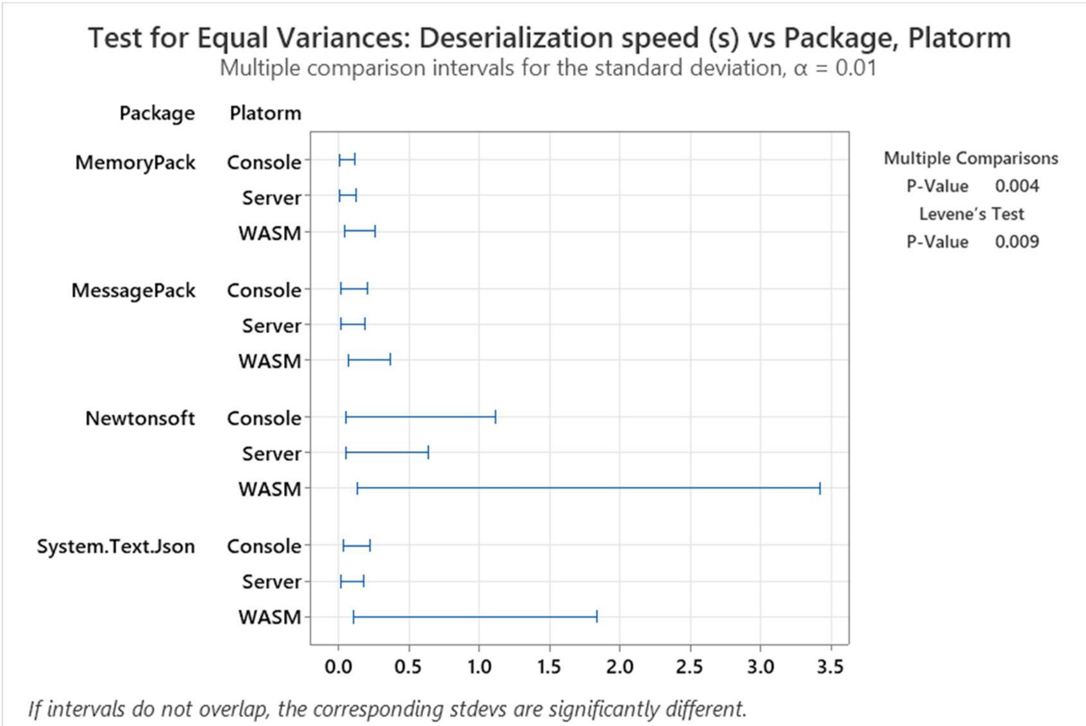


Figure 2

Equal variance test for deserialization speed



Levene's Test indicated that the variances of our samples are different meaning we need to create a new model for our data. Therefore, we will apply a transformation by taking log of the response to fit the data to the model. The results for the new model variance test are in figure 3 and 4.

Figure 3

Equal variance test for log(serialization speed)

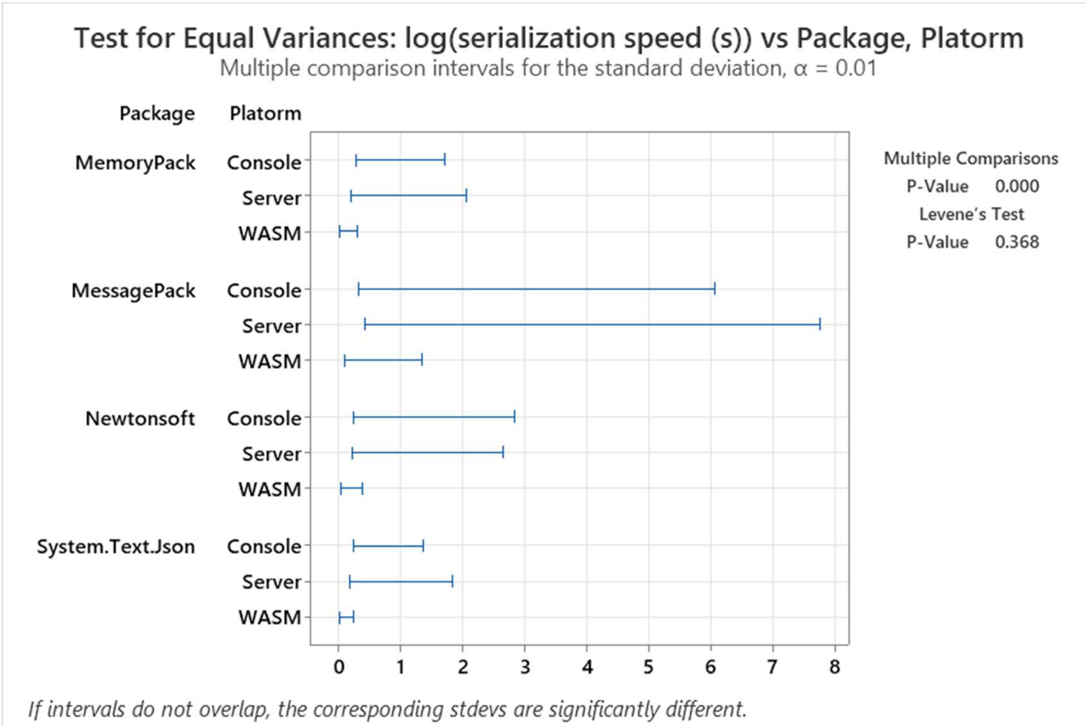
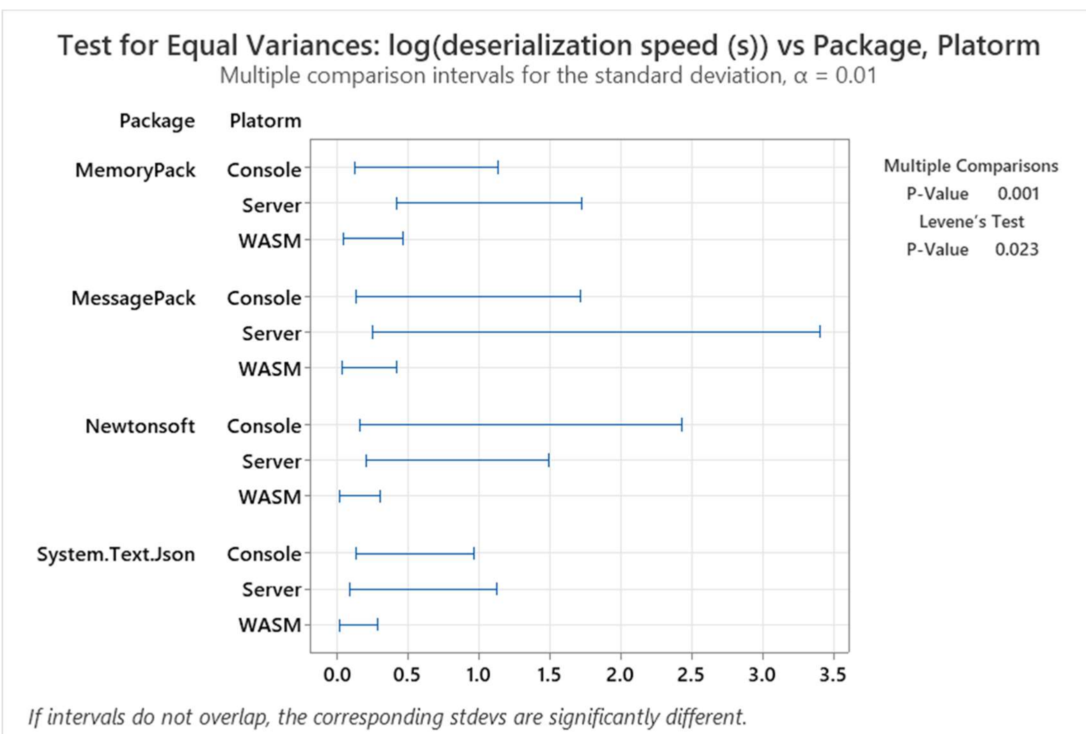


Figure 4

Equal variance test for log(deserialization speed)



The following residual plots (figures 5 and 6) also support our first 2 assumptions of normality and independent observations.

Figure 5

Residual plots for the log(serialization speed)

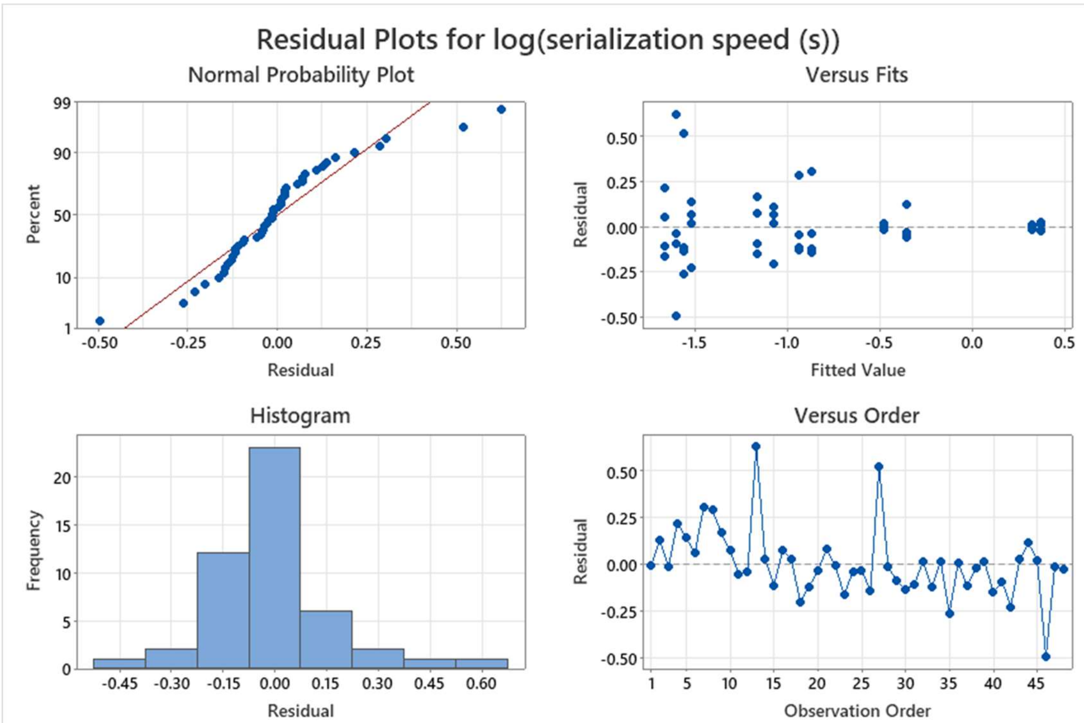
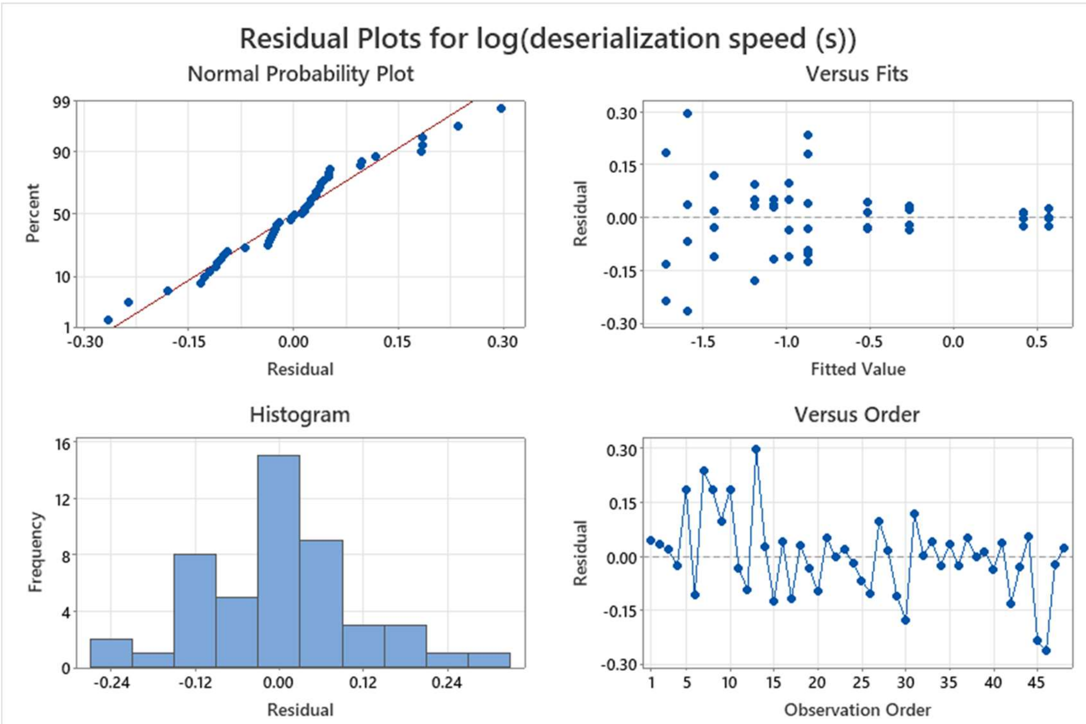


Figure 6

Residual plots for the log(deserialization speed)



With our assumptions met, we could now apply an ANOVA analysis to test for significant differences in serialization and deserialization speeds. Tables 5 and 6 show the results of the ANOVA test.

Table 5

ANOVA analysis for the log(serialization speed)

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Platform	2	17.0321	8.51604	194.68	0.000
Package	3	5.0485	1.68285	38.47	0.000
Platform*Package	6	0.2114	0.03523	0.81	0.572
Error	36	1.5748	0.04374		
Total	47	23.8668			

Table 6

ANOVA analysis for the log(deserialization speed)

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Platform	2	17.5310	8.76548	554.63	0.000
Package	3	5.4816	1.82719	115.61	0.000
Platform*Package	6	0.5391	0.08985	5.69	0.000
Error	36	0.5690	0.01580		
Total	47	24.1206			

The lack of significance in the interaction effects of platform and package for the log(serialization) means that we can remove it from the model. Therefore, our final ANOVA analysis for log(serialization) is shown in table 7. Our new model for this data is below.

$$y_{ijk} = \mu + \tau_i + \beta_j + \varepsilon_{ijk} \begin{cases} i = 1, 2, \dots, a \\ j = 1, 2, \dots, b \\ k = 1, 2, \dots, n \end{cases} \tag{6}$$

μ - grand mean
τ_i - ith level of package factor
β_j - jth level of platform factor
ε_{ijk} - random error, $\sim N(0, \sigma^2)$

This model still has the following constraints.

$$\sum_i \tau_i = 0 \tag{7}$$

$$\sum_j \beta_j = 0 \tag{8}$$

Table 7

ANOVA analysis for the log(serialization speed)

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Platorm	2	17.032	8.51604	200.24	0.000
Package	3	5.049	1.68285	39.57	0.000
Error	42	1.786	0.04253		
Total	47	23.867			

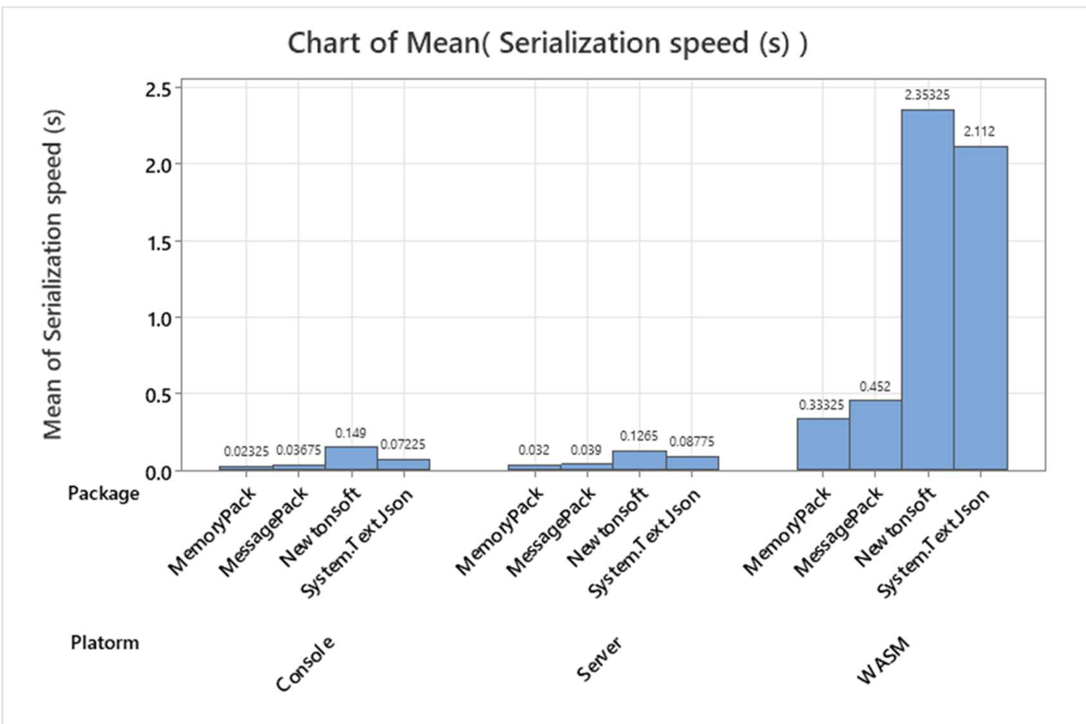
The low p-values indicate that there are significant changes in log(serialization) and log(deserialization) speed for different software packages and platforms. However, there are only significant differences in the combination of software packages and platforms for deserialization.

Results

Figure 7 and figure 8 provide helpful representations of the speed of serialization and deserialization for easy comparison.

Figure 7

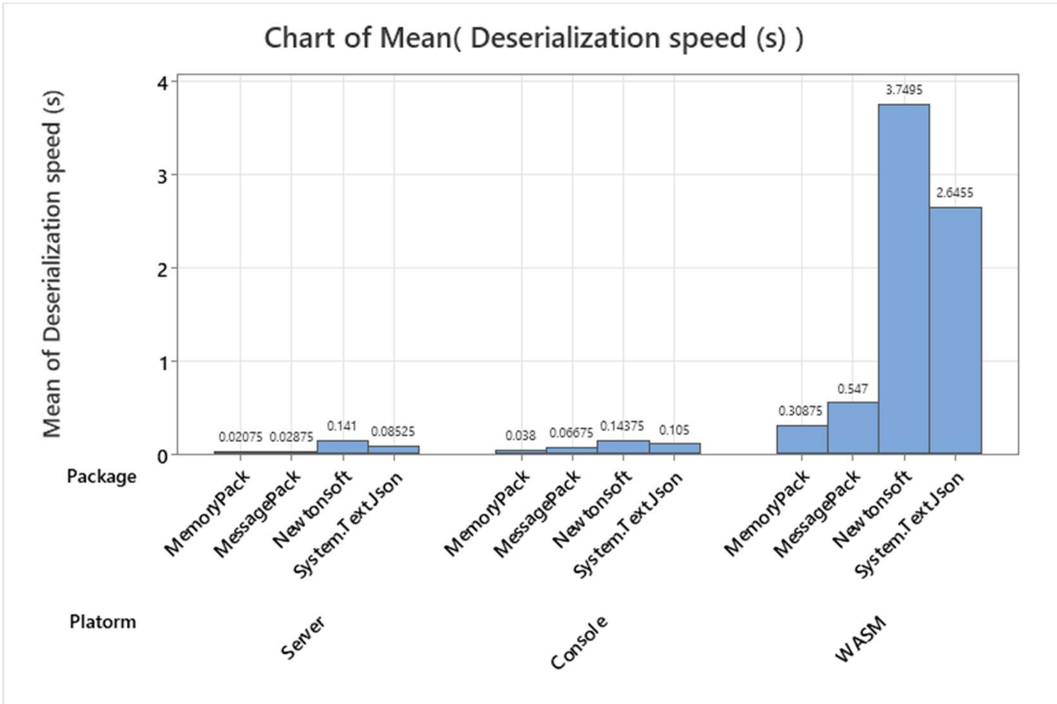
Bar chart for the mean serialization speed



For serialization, the fastest combination is using MemoryPack on a console app, which took an average of 0.02325 seconds. The slowest combination is using Newtonsoft on a WASM application, which took an average of 2.35325 seconds.

Figure 8

Bar chart for the mean deserialization speed



For deserialization, the fastest combination is using MemoryPack on a server app, which took an average of 0.02075 seconds. The slowest combination is using Newtonsoft on a WASM application, which took an average of 3.7495 seconds.

Conclusions

Our factorial design analysis showed significant differences between the platform, package, and interaction effect for both serialization and deserialization times. Our results suggest that developers should carefully consider the choice of platform and package when working with serialization and deserialization tasks.

Overall, we found that MemoryPack had the lowest mean serialization and deserialization times, followed by MessagePack, Newtonsoft, and System.Text.Json, respectively. The WASM platform had the longest mean serialization and deserialization times, followed by the Console and Server platforms. Additionally, the interaction effect between platform and package was significant for the deserialization time, indicating that the performance of each package is affected differently by the platform it runs on.

These findings provide useful insights for developers and software engineers who work with serialization and deserialization tasks. These results suggest that choosing the right combination of platform and package can have a significant impact on the efficiency of these tasks. Further research can investigate the performance of other platforms and packages and explore their interactions in more detail.

Appendix

Table 1

Observations of serialization time

Platform	MemoryPack	MessagePack	Newtonsoft	System.Text.Json
Console	0.036	0.091	0.273	0.101
	0.025	0.020	0.102	0.083
	0.015	0.015	0.124	0.056
	0.017	0.021	0.097	0.049
Server	0.042	0.105	0.225	0.100
	0.036	0.023	0.105	0.089
	0.018	0.020	0.089	0.053
	0.032	0.008	0.087	0.109
WASM	0.324	0.597	2.492	2.061
	0.322	0.391	2.417	2.050
	0.336	0.402	2.229	2.174
	0.351	0.418	2.275	2.163

Table 2

Observations of deserialization time

Platform	MemoryPack	MessagePack	Newtonsoft	System.Text.Json
Console	0.035	0.081	0.234	0.129
	0.029	0.043	0.126	0.116
	0.039	0.070	0.108	0.080
	0.049	0.073	0.107	0.095
Server	0.029	0.051	0.206	0.092
	0.029	0.022	0.109	0.064
	0.014	0.028	0.101	0.090
	0.011	0.014	0.148	0.095
WASM	0.340	0.588	3.971	2.629
	0.320	0.504	3.764	2.746
	0.288	0.521	3.714	2.491
	0.287	0.575	3.549	2.716

Table 3

Mean of observed serialization time

Platform	MemoryPack	MessagePack	Newtonsoft	System.Text.Json
Console	0.02325	0.03675	0.14900	0.07225
Server	0.03200	0.03900	0.12650	0.08775
WASM	0.33325	0.45200	2.35325	2.11200

Table 4

Mean of observed deserialization time

Platform	MemoryPack	MessagePack	Newtonsoft	System.Text.Json
Console	0.03800	0.06675	0.14375	0.10500
Server	0.02075	0.02875	0.14100	0.08525
WASM	0.30875	0.54700	3.74950	2.64550

Figure 7

Bar chart for the mean serialization speed

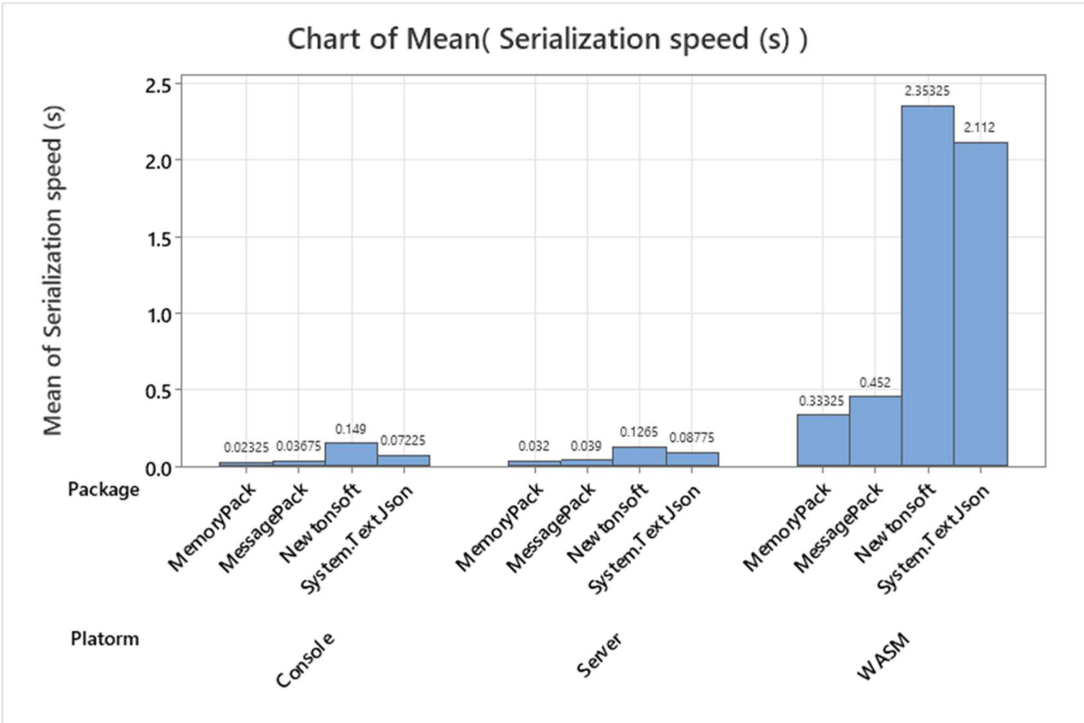


Figure 8

Bar chart for the mean deserialization speed

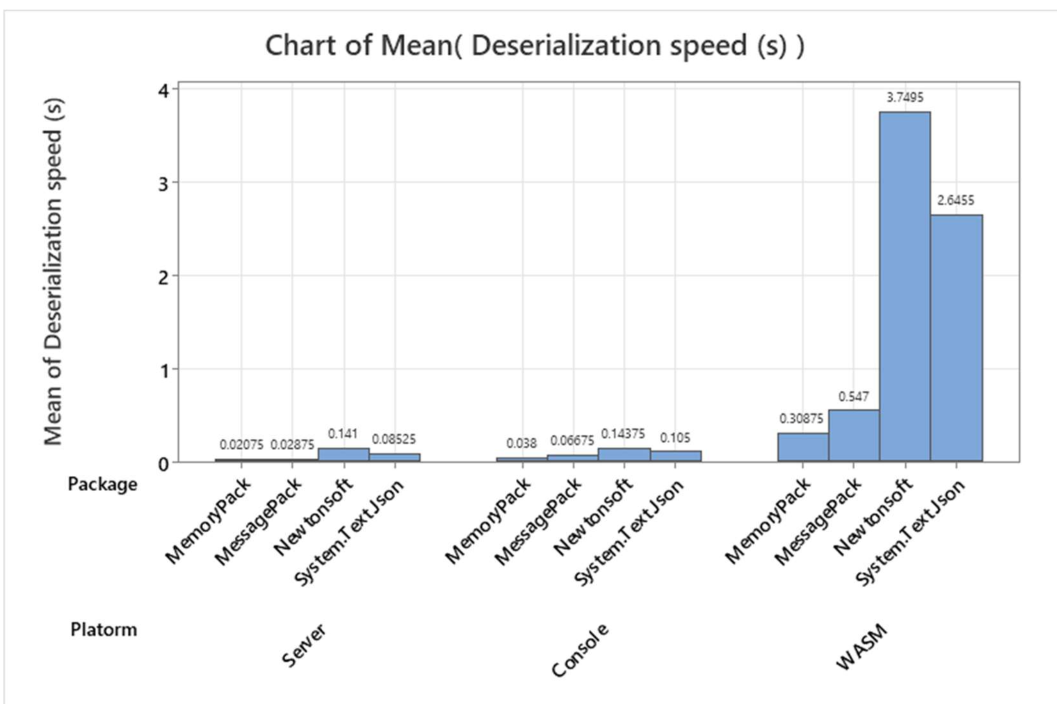


Figure 9

Bar chart for the mean serialization speed in console app

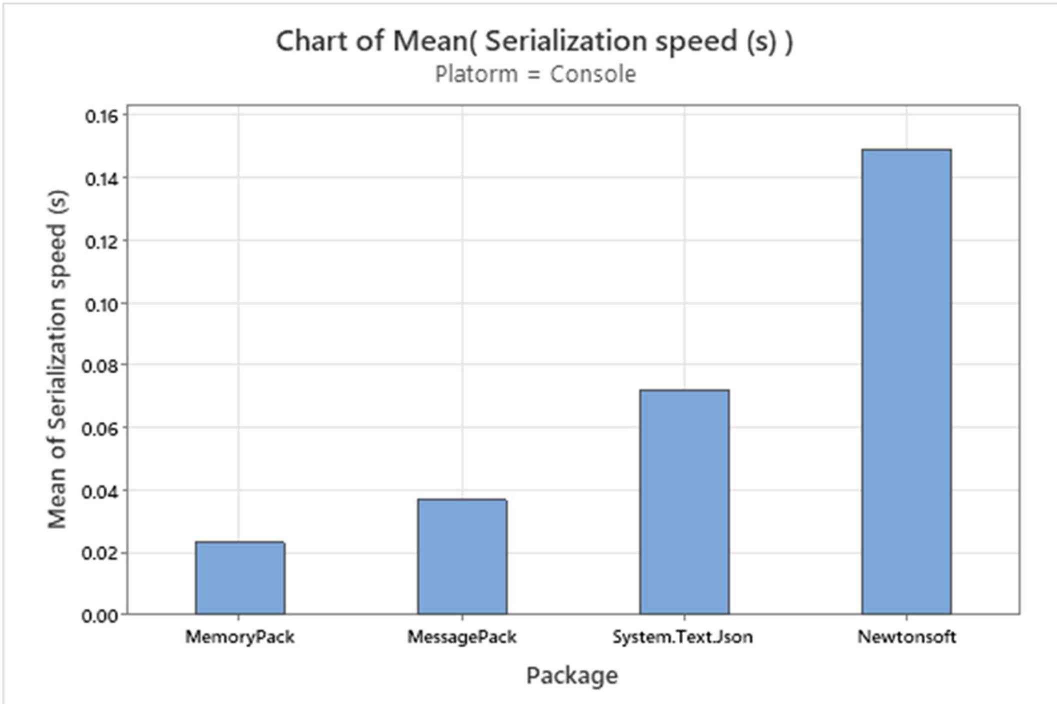


Figure 10

Bar chart for the mean serialization speed in server app

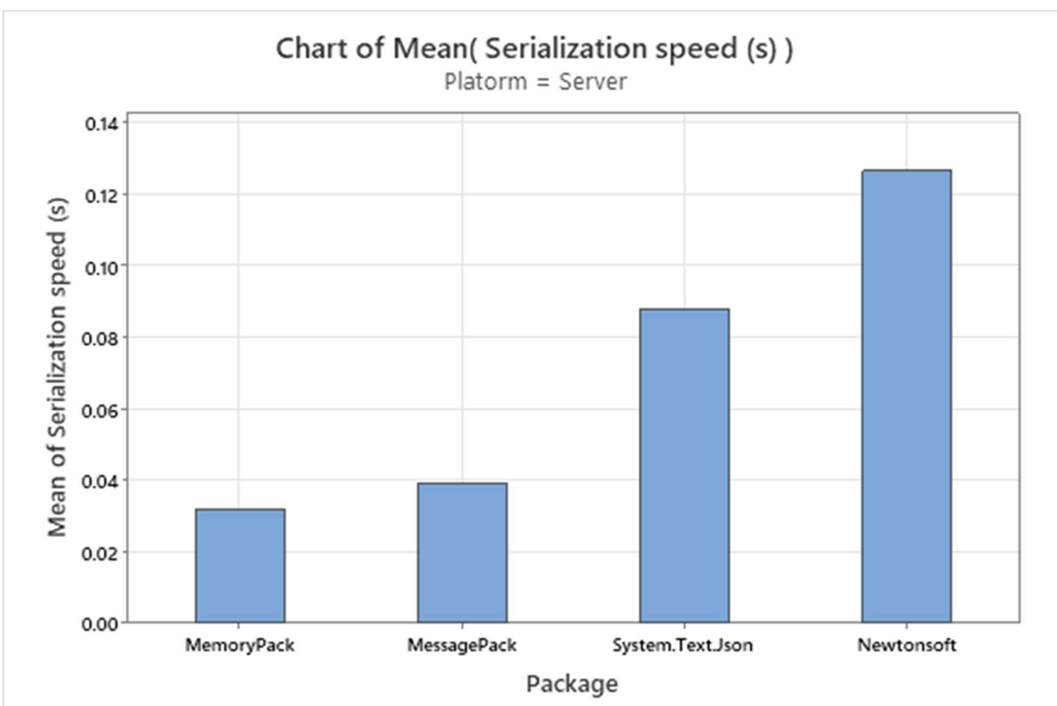


Figure 11

Bar chart for the mean serialization speed in wasm app

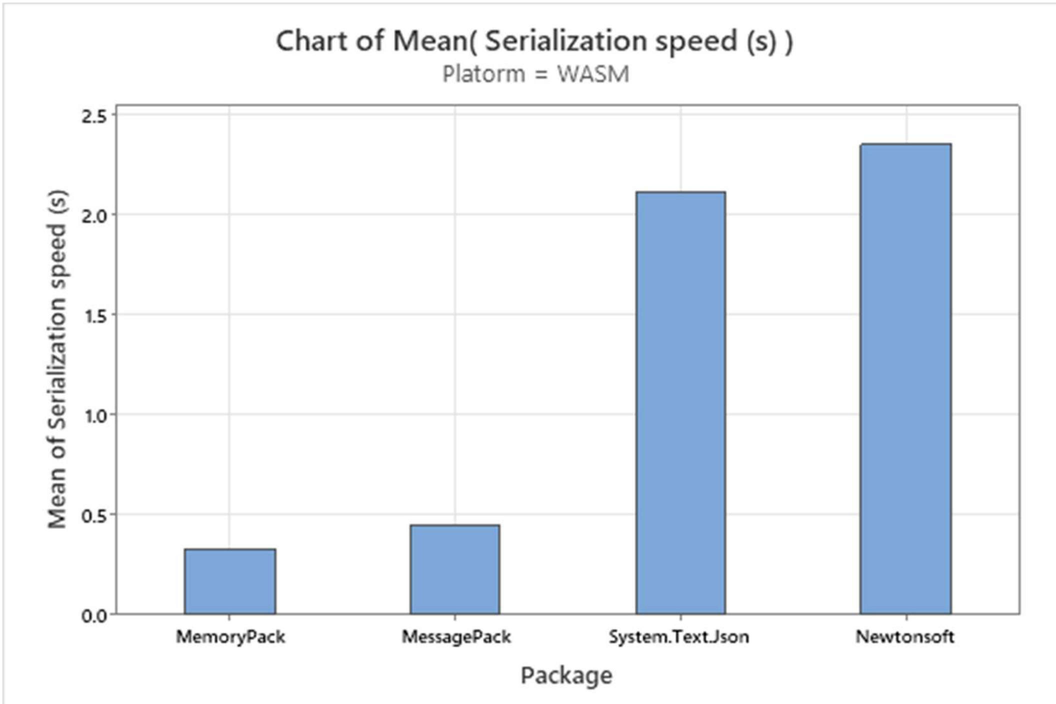


Figure 12

Bar chart for the mean deserialization speed in console app

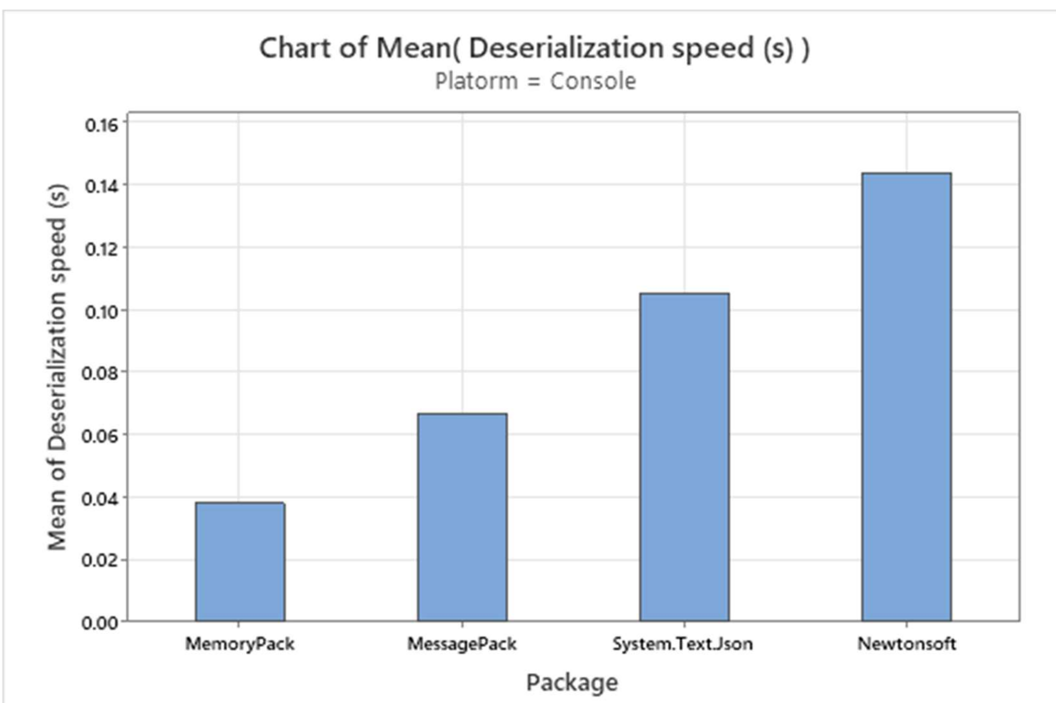


Figure 13

Bar chart for the mean deserialization speed in server app

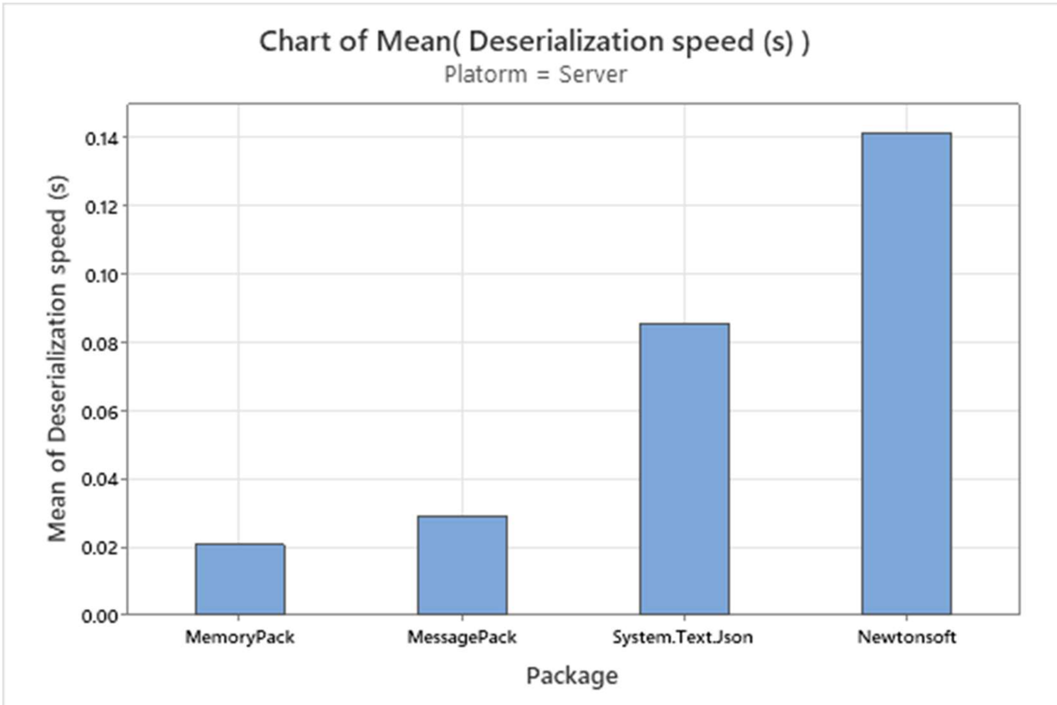


Figure 14

Bar chart for the mean deserialization speed in wasm app