Data Representation and Serialization Formats

Lino Varela
University of Coimbra, DEI
Email: linovarela@student.dei.uc.pt
Nuno Carvalho
University of Coimbra, DEI
Email: nunascimento@student.dei.uc.pt

Abstract—**Data serialization is a critical process in distributed systems and microservices, that allows the exchange of information between different services. This information can be in different formats like XML or JSON, which are text-based or formats like Protocol Buffers that are binary. A big problem can be determining which of these formats can give the best results for serializing and deserializing objects in a real-world scenario. This paper will explain and try to answer the question of whether JSON or XML is more efficient in serializing and deserializing data, providing the results of experiments and recommendations for data representation choices. For this, simple scripts were developed to serialize and deserialize data in both formats. These were used with different sizes of data to evaluate the time needed for the operation and take conclusions. With the results, we will answer the question previously mentioned.**

I. Introduction

With the increasing usage of distributed systems and microservices, the choice of data serialization formats has become an important factor in system performance. There are several different formats to transform the data for this communication. We investigated the performance of XML and JSON for this process of serialization and deserialization, more specifically the speed of this operations.
This project will evaluate if "Json outperforms XML in terms of Data serialization and deserialization in a microservices environment", being this hypothesis explored and tested to be verified using the programming language *Java* to conduct these tests where different sizes of data used to get the results.

II. Concepts

Some important concepts were needed to develop the project.

- Data Serialization is one of the most important concepts for this project, it is the process of converting complex data or objects into a format that can be easily transmitted, stored, or reconstructed later, enabling exchange of information between different components or applications over a network.
- Serialization Formats were developed to handle the data transformation. They define the structure and rules for converting data into a standardized format, being crucial for enabling transparent communication between systems that may use different programming languages or platforms.
- XML (Extensible Markup Language) is a text-based format to represent complex data hierarchies in a structured way. The data can be represented as elements and attributes (it is object-oriented), being this format readable for humans and can be automatically validated. A concern with this format is the overhead in terms of file size and processing time when applied to high-performance systems, particularly as data volumes increase.
- JavaScript Object Notation, also known as JSON is another data representation format easy for humans to read and write. It has become the preferred format for many different services due to its simplicity, lightweight structure and the fact that is compatible with most programming languages. This format has limited data type support, it lacks some of the capabilities that XML might provide like supporting comments within data or multi-line strings. In terms of extensibility, JSON is not, because it doesn't need to be and it has the same interoperability potential of XML.
- JSON and XML, even though they are both easily readable by both humans and machines, JSON is easier to read and much simpler than XML. Another important aspect is that XML is document-oriented while JSON is data-oriented, being mapped more easily on object-oriented systems.

III. Problem Statement

The amount of information microservices need to handle is always increasing, and the efficiency of data serialization becomes more and more crucial for system performance and stability. In an environment that demands frequent data exchange, choosing the most optimal serialization format can have a significant impact. An important challenge is knowing which serialization format can perform better when handling different data sizes. With larger datasets, the differences in efficiency, processing speed, and overhead between the formats become more pronounced. This work aims to explore how XML and JSON, two of the most commonly used serialization text-formats, compare when serializing and deserializing datasets of increasing sizes

IV. Experimental Setup

For the experimental tests, scripts were developed to serialize and deserialize data in both formats being analyzed. The scripts were implemented in *Java* and executed in a consistent and controlled environment to ensure accuracy and repeatability. The experiments were done on a computer with the following specifications:

- Computer model - Aspire A515-55g
- Processor - i7-1065G7
- RAM - 8 GB

The data used in the tests included different sizes of datasets, specifically designed to simulate realistic scenarios, ranging from small to large numbers of objects.
The objects are based on a simple school management system, with different schools and students, each with their specifications and information.

A. Dependencies

As said, *Java* was used for the scripts, *maven* dependencies were used, such as:

- JAXB API Dependencies: These are required for XML binding and marshalling/unmarshalling using Jakarta XML Binding (JAXB).
- JSON (Gson) Dependency: Gson, for JSON serialization and deserialization, simplifying JSON data handling in *Java*.

These dependencies help ensure the code's functionality for handling both XML and JSON is efficient.

B. Serialization

- **XML:** A JAXBContext was created for the binding functionality, and a Marshaller was used to convert *Java* objects into XML. The data, structured as a list of school objects, was wrapped in a container class and serialized as a single XML file.
- **JSON:** For JSON, it was used the GSON library. The list of school objects was serialized into a JSON string, which was then written to a file.

C. Deserialization

Deserialization also varied between the two formats:

- **XML Deserialization:** The XML data was deserialized by creating a JAXBContext and using an unmarshaller to transform the XML back into *Java* objects.
- **JSON Deserialization:** The GSON library was used for JSON deserialization. The JSON file was read into a string and parsed back into a list of school objects using the Gson parser.

Both of this processes were measured in nanoseconds, using the *Java* method *nanotime()*.

V. Results

XML		
Number of objects	**Serialization time**	**Deserialization time**
100	3370000	16239900
500	10184600	35190900
1000	10365100	32933900
2500	20868800	42307500
5000	23857000	58638400
7500	35930200	55950100
10000	33869200	69754600
15000	34565700	94012900

JSON		
Number of objects	**Serialization time**	**Deserialization time**
100	2804700	1689300
500	7786300	6647600
1000	10867900	7616600
2500	18284500	6491700
5000	22733300	12494700
7500	30604100	16800000
10000	38172000	19289100
15000	45816100	19607500

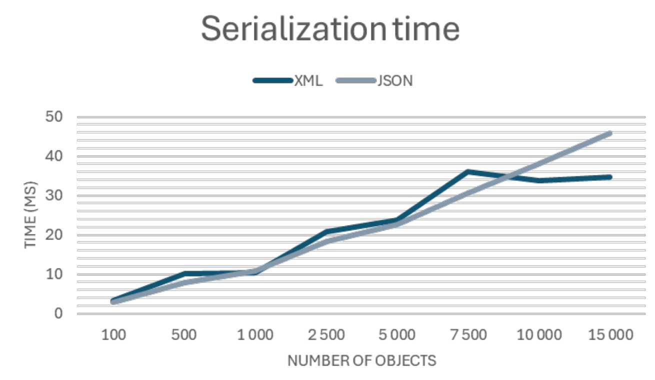

Fig. 1. Serialization times

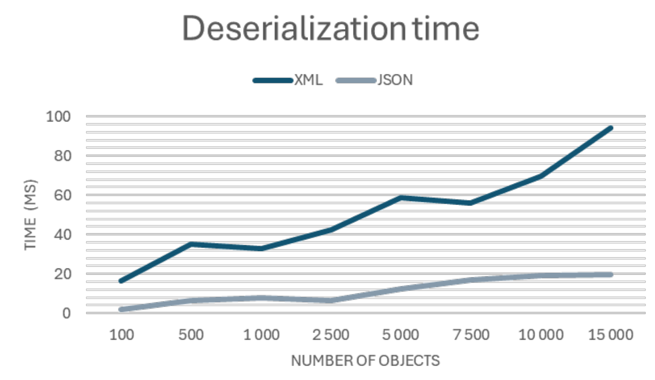

Fig. 2. Deserialization times

VI. Discussion

With the results obtained, it is possible to get clear insights regarding the performance differences between the two formats being studied, in both serialization and deserialization. For serialization, the results show a generally expected increase in time as the number of objects grows. JSON performs more consistently, with a linear growth in serialization time across the various dataset sizes. In contrast, XML exhibits a less predictable pattern, particularly with a notable spike at 7,500 objects. This anomaly may happen because XML might not have a consistent pattern when the volume of data increases or because of system-level factors. Discarding this, XML's serialization time shows similar growth trends as JSON up until around 10,000 objects, at which point XML becomes faster, which might be unexpected, outperforming JSON in larger datasets. This suggests that XML is better suited for scenarios involving large-scale data serialization.

On the other hand, When it comes to deserialization, JSON consistently outperforms XML across all the tested sample sizes. JSON's ability to deserialize data more quickly makes it a better choice for applications where fast data retrieval is essential. XML, while not as fast as JSON in deserialization, still scales predictably as dataset sizes increase, but the performance gap becomes more pronounced with larger amounts of data. The difference between the two formats is more noticeable when the number of objects is above 5,000, where XML deserialization times start to rise more sharply, suggesting that JSON would be the more efficient option for applications where deserialization speed is critical.

VII. Conclusion

In conclusion, with the experiments tests that were developed and used, a clear comparison of the time efficiency of XML and JSON in serialization and deserialization across varying dataset sizes was made. While both formats show an increase in time proportional to the growth in data size, their performances diverge based on the operation. JSON demonstrates superior performance in deserialization, making it the more suitable choice for systems requiring quick data retrieval or frequent deserialization operations. On the other hand, XML shows advantages in serialization efficiency when dealing with larger datasets, particularly beyond 10,000 objects, where it surpasses JSON in speed. Overall, JSON is better suited for scenarios with frequent and time-sensitive deserialization, while XML might be a good option if the dataset size grows and serialization efficiency takes precedence.

References

[1] Baeldung, "Working with json in java," https://www.baeldung.com/java-org-json, 2024.
[2] F. Araújo, "Json," Lecture notes, 2015, 2015-24.
[3] ——, "Basic xml," Lecture notes, 2015, 2015-23.
[4] T. Point, "Maven tutorial," https://www.tutorialspoint.com/maven/, 2024.
[5] Mkyong, "Jaxb hello world example," https://mkyong.com/java/jaxb-hello-world-example/, 2024.
[6] I. Cloud, "Json vs xml: What is the difference?" https://www.imaginarycloud.com/blog/json-vs-xml, 2024.