**Deno vs Rust: How faster is machine code compared to interpreted code for JWT sign & verify?**

[[Mayank Choubey](https://medium.com/@choubey?source=post_page-----dab52bc60f5b--------------------------------)](https://medium.com/@choubey?source=post_page-----dab52bc60f5b--------------------------------)

[[Tech Tonic](https://medium.com/deno-the-complete-reference?source=post_page-----dab52bc60f5b--------------------------------)](https://medium.com/deno-the-complete-reference?source=post_page-----dab52bc60f5b--------------------------------)

[Mayank Choubey](https://medium.com/@choubey?source=post_page-----dab52bc60f5b--------------------------------)

·

Follow

Published in

[Tech Tonic](https://medium.com/deno-the-complete-reference?source=post_page-----dab52bc60f5b--------------------------------)

·

3 min read

·

Nov 18, 2023

7

1

**Introduction**

In this upcoming series, we’ll delve deeper into the performance benchmarking of a widely used but modest-sized functionality: the signing (generation) and verification of JSON Web Tokens (JWT). JWT is an open standard that outlines a concise and self-contained method for securely transmitting information in the form of a JSON object between different parties. The reliability of this transmitted information is ensured by digital signatures. JWTs can undergo signing using a secret, employing the HMAC algorithm, or by utilizing a public/private key pair through RSA or ECDSA.

Throughout this series, we will conduct benchmark tests for JWT signing, generation, and verification across a spectrum of programming languages and environments, including Node.js, Deno, Bun, Python, Go, Rust, and Java. Our approach involves employing the most commonly utilized JWT framework within each respective technology. It’s crucial to recognize that the benchmarking process looks to be comparing the performance of frameworks, rather than evaluating the programming languages themselves. But we’re aware that these frameworks don’t operate on their own. They make use of language features, so when we compare them, we’re also evaluating the performance of programming languages.

In this particular article, we aim to compare the performance of Deno and Rust specifically for JWT functionality. This is not even an apple to orange comparison. Interestingly, Deno itself is written in Rust. The anticipated outcome is already established: Rust is expected to outperform Deno by a big landslide. When it comes to a demanding task such as JWT, the disparity between machine code and interpreted code is likely to be significant. However, the precise extent of this speed difference is the focal point of our inquiry.

**Setup**

All assessments were conducted on a MacBook Pro M2 equipped with 16 gigabytes of RAM.

The software versions are:

* Deno v1.38.1
* Rust v1.73.0

To ensure a sufficient degree of randomness in our evaluations, we incorporated 100,000 randomly generated email IDs as the **sub** field in the JSON Web Token (JWT).

The testing procedure was executed sequentially, encompassing 1 million rounds or iterations. Each round comprised the following steps:

1. Selection of the next email address.
2. Creation of a payload containing sub, iat, and exp fields.
3. Signing and generating a compact JWT.
4. Running the JWT through the verification process.
5. Retrieval of the payload and verification of the associated email.

For each contender, we collected two key measurements:

1. The time required to complete 1 million rounds.
2. The number of operations per second (OPs), where an operation is broadly defined as the combined processes of signing and verifying the JWT.

**Code**

The code for both Deno and Rust applications as follows:

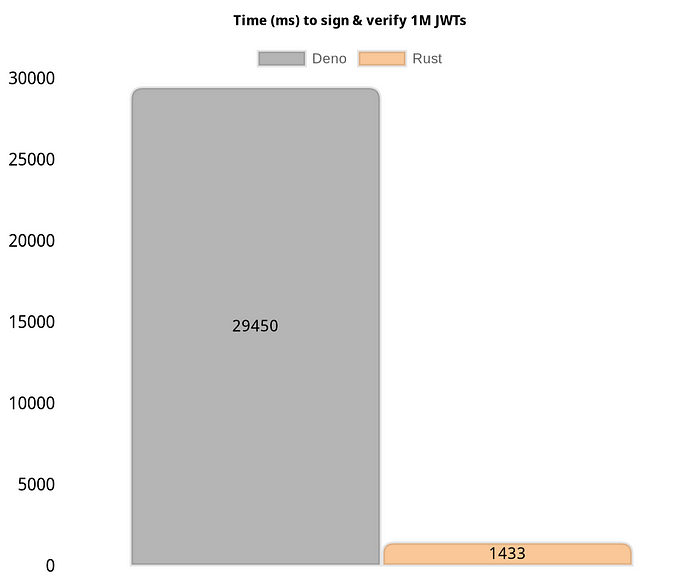
**Deno**

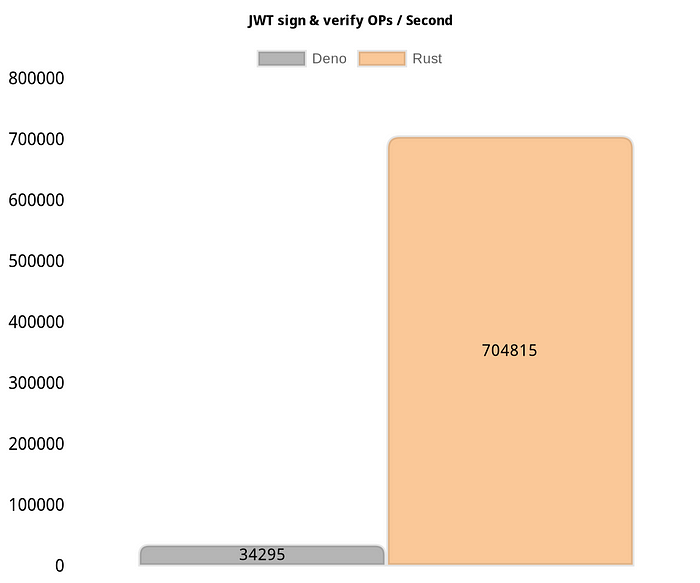
**Rust**

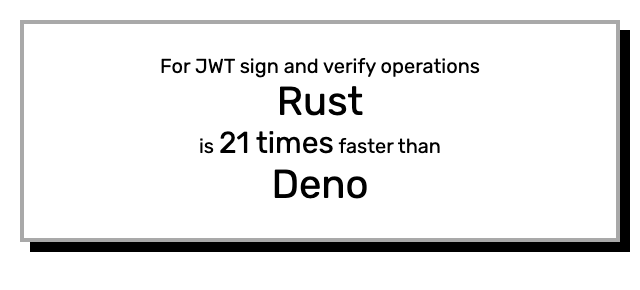
Rust application has been built in release mode.

**Results**

The results for 1M iterations in chart form are as follows:







**Analysis**

The results we discovered are truly remarkable. We anticipated that machine code (Rust) would outperform interpreted code (Deno), but the magnitude of the distinction exceeded what we had anticipated. Rust manages to complete 1 million rounds in a mere 1.5 seconds, while Deno takes around 29 seconds — a significant difference. It’s crucial to emphasize that these outcomes are specific to this particular scenario, and variations in performance may arise in different situations.

Thanks for reading this article!

[Deno](https://medium.com/tag/deno?source=post_page-----dab52bc60f5b---------------deno-----------------)

[Rust](https://medium.com/tag/rust?source=post_page-----dab52bc60f5b---------------rust-----------------)