Rust vs C++: Perforamnce in an extended sieve of Eratosthenes, an approximation of π and and approximation of Euler's Number

Zachary Meyner

Rust is a programming language maintained by Mozilla that is designed to maximize speed—in terms of time spent coding and execution—as well as safety. Rather than a garbage collector or manually managing memory safety of Rust is achieved through its system of ownership—a set of rules checked at compile time to ensure safety [1]. The rules of ownership are:

- Each value has an owner,
- Each value can only have one owner, and
- When the owner goes out of scope the variable is deleted [1].

Rust also employs a system of borrowing for referencing. When a variable needs to be used in a different part of code, say a function,

- The variable is borrowed by that function via a reference and does not own that variable,
- The function cannot return that reference, and
- The reference is deleted at the end of the function call because of ownership [1].

Rust also includes a very robust package manager and database, and build tool called "Cargo" [2]. Cargo's package manager and crates.io makes community support for any features not included in the standard library to be maintained. Rust's technology has allowed it to rank as the most popular language on Stack Overflow developer survey for seven years [3]. The language has also amassed a large dedicated fanbase, including a discord server with 40,000+ users [4].

Rust's potential in speed and lines of code (LOC) written helped raise the popularity of the language and brought about speed tests of the language. It has been run against C in the embedded software and determined to be a viable alternative [5]. The safety benefits in systems programming has been explored [6]. Rust has also been compared to C, Fortran, and Java for speed and effort in parallel architectures, and found to be as fast as the fastest languages with the lowest amount of effort when programming [7] [8].

The Sieve of Eratosthenes finds all prime numbers $p < n$ by starting at the first prime (normally 3) p_0 and multiplying by $k \leq \sqrt{n}$ where $k \in \mathbb{N}$ and marking resultant numbers as not prime. When all k have been multiplied by p_0 the steps are repeated on the next unmarked number p_1. The extended Sieve of Eratosthenes divides the range of the sieve into segments of size $\sqrt{n} + 1$ and mark prime numbers in the respective ranges one segment at a time. Dave Plummer has run a Sieve of Eratosthenes "race" on many programming languages [9]. Plummer's results show a basic Sieve as opposed to an extended sieve, and rather than testing speed, Plummer tested how many times the sieve can go up to a number in a certain amount of time. The sieve of Eratosthenes gives an excellent test of the languages standard library vector implementations, and ease of writing array code in each langauge.

Approximating π has been one of the longest problems in Mathematics, dating back to Ancient Egypt [10]. Methods for approximating π have become quite computationally efficient since then and eventually lead to Ramunujan-type approximation by the Chudnovsky brotherswhich was used to break the the world record for computing π [11]. Approximating Euler's Number can be done through the Taylor Series derived from the equation e^x. Both these numerical approximations provide a great way to compare the compilation and LOC efficiency of Rust and C++, and external supported packages in each langauges.

References

[1] S. Klabnik and C. Nichols, *The rust programming language*. No Starch Press, 2023. [Online]. Available: https://doc.rust-lang.org/book/title-page.html

[2] "The cargo book." [Online]. Available: https://doc.rust-lang.org/cargo/index.html

[3] "Stack overflow developer survey 2022," Jun 2022. [Online]. Available: https://survey.stackoverflow.co/2022/

[4] "Rust programming language community server discord server." [Online]. Available: https://discord.com/invite/rust-lang-community

[5] N. Borgsmüller, "The rust programming language for embedded software development," Ph.D. dissertation, Technische Hochschule Ingolstadt, 2021.

[6] A. Balasubramanian, M. S. Baranowski, A. Burtsev, A. Panda, Z. Rakamarić, and L. Ryzhyk, "System programming in rust: Beyond safety," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, ser. HotOS '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 156–161. [Online]. Available: https://doi.org/10.1145/3102980.3103006

[7] M. Costanzo, E. Rucci, M. Naiouf, and A. D. Giusti, "Performance vs programming effort between rust and c on multicore architectures: Case study in n-body," in *2021 XLVII Latin American Computing Conference (CLEI)*, 2021, pp. 1–10. [Online]. Available: https://doi.org/10.48550/arXiv.2107.11912

[8] H. Heyman and L. Brandefelt, "A comparison of performance & implementation complexity of multithreaded applications in rust, java and c++," B.S. thesis, KTH Royal Institute of Technology, 2020. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-280110

[9] D. Plummer, "Primes," 2023. [Online]. Available: https://github.com/PlummersSoftwareLLC/Primes

[10] D. M. Burton, *Egyptian Geometry*. McGraw Hill, 2011, pp. 55–55.

[11] B. Lynn, "Ramanujan's formula for pi." [Online]. Available: https://crypto.stanford.edu/pbc/notes/pi/ramanujan.html