Zachary Meyner

Performance of fast sorting algorithms in Rust and C++.

References

[1] 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

Have yet to read

[2] W. Bugden and A. Alahmar, "Rust: The programming language for safety and performance," *arXiv preprint arXiv:2206.05503*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2206.05503

Overview: This article presents an summary of the Rust Programming langauge and analyzes the features that could be responsible for its rise in popularity. Rust achieves this through its high performance combined with its excellent memory safety features. Despite this Rust does not see as much use as older performant langauges like C and C++ because of how new it is, even with its quickly growing follower-base. Thus, they recommend that more work be done to increase use of Rust among software engineers.
Evaluation: A lot of good information in this article; although, pretty opinionated at times. Not much in terms of benchmarks or empiracal ways to measure the worth of a language.
Comments: There is some nice to have info on Rust that I

wish were here, like what a ”panic” in rust actually does, and more info on ownership and the borrow-checker.

[3] 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

Overview: This article compares the performance of parallel computing speed of Rust and C in a gravitation N-Body problem. It compares many different variants of the Rust implementation, including use of different allocators, mathematical optimizations, use of different iterators, vectorizing with AVK-512, and block processing to the most optimized C implementaiton in 32 and 64 bit float implementation. They discovered that C was faster in 32 bit float simulations, but the langauges were equivalent in 64 bit float simulations. They also considered how the work done to implement the N-Body problem, finding it easier and faster in Rust once the langauge was understood. They suggest future work be done on other computationally intensive problems with different characteristics, including other programming langauges, and considering other CPU architectures.
Evaluation: This is good; however, it focuses a lot on the optimization and consequences of having to deal with floats.

[4] 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

Overview: This article compares multithreaded performance in a key-value pair database developed in Rust, C++, and Java. Their implementation of a key-value pair was done through a hashmap. Performance tests were performend on read and writes to the database. They varied the number of read and

writes done on the datase, as well the number of cores available to use. Read and write performance in Rust and C was about equivalent while Java was always the slowest. They also took into account how many lines of code each langauge used to create the database, with Rust being less than Java, and signifigantly less than C++.
Evaluation: This is a nice and simple article. It is very focused on the analysis of key-value db performance. A big limitation is that the hardware used for testing is over a decade old.

[5] M. Sudwoj, "Rust programming language in the high-performance computing environment," B.S. thesis, ETH Zurich, 2020. [Online]. Available: https://doi.org/10.3929/ethz-b-000474922

Overview: This article compares the performance of Rust, C++, and Fortran implementations of the fourth-order numerical diffusion equation. They test solutions to this differential equation under many different conditions, techonologies, and compliers for each language when possible. They conclude that Rust is faster or just as fast as Fortran and C++. Additionally they find the features and safety of the langauge to be useful.
Evaluation: This article has a lot of useful stuff, so much that it is hard to read it. Each test only being run once is not great, the hardware they run the tests on is quite old as well. It feels as though they are trying to do too much with this paper with how much was considered and tested.
Comments: The performance tests here need to be run more times on newer hardware.

[6] F. Wilkens, "Evaluation of performance and productivity metrics of potential programming languages in the hpc environment," B.S. thesis, University of Hamburg, 2015.

Haven't read yet