The Journal of Open Source Software

RustSASA: A Rust Crate for Accelerated Solvent Accessible Surface Area Calculations

Maxwell J. Campbell[1]

1 University of California, San Francisco, United States

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Solvent accessible surface area (SASA) calculations are fundamental for understanding protein structure, function, and dynamics in computational biology. These calculations quantify the surface area of biomolecules accessible to solvent molecules, providing insights into protein folding, stability, and intermolecular interactions. The Shrake-Rupley algorithm has served as the standard for SASA calculations since 1973, but existing implementations often become computational bottlenecks when analyzing large protein datasets. As proteomics datasets continue to grow exponentially—with initiatives like AlphaFold producing hundreds of millions of predicted protein structures—the need for efficient SASA calculation tools has become increased dramatically. RustSASA addresses this challenge by providing a high-performance implementation of the Shrake-Rupley algorithm written in pure Rust, delivering a 7× speed improvement over Freesasa while maintaining calculation accuracy and providing interfaces for multiple programming languages.

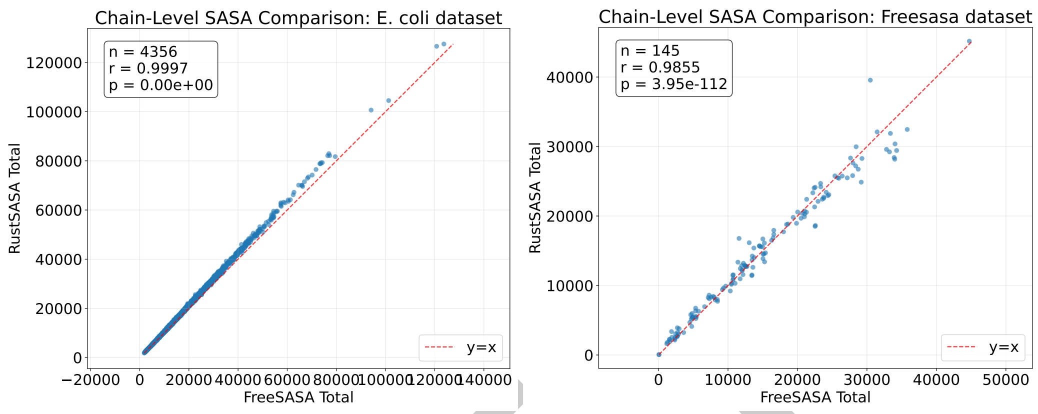
Statement of need

Current SASA calculation tools represent a significant computational bottleneck in structural biology workflows, particularly for molecular dynamics simulations and high-throughput analyses. Popular implementations such as those in Biopython and Freesasa, while accurate, become prohibitively slow when processing large protein datasets.

RustSASA addresses this performance gap by leveraging Rust's zero-cost abstractions and memory safety guarantees. Benchmarking on representative protein structures demonstrates that RustSASA achieves a 7× improvement over Freesasa and a 46× improvement over Biopython implementations. This performance advantage reduces computational costs for high-throughput structural analyses and makes large-scale comparative studies feasible. Furthermore, RustSASA's multi-language support (Rust and Python) and command-line interface ensure broad accessibility across the computational biology community.

Results

Calculation Quality



To evaluate the accuracy of RustSASA calculations, we compared results to Freesasa (Mitternacht, 2016) on both the predicted E. coli proteome from AlphaFold (Jumper et al., 2021) and the Freesasa evaluation dataset.

RustSASA produces SASA values that closely match those from Freesasa, achieving Pearson correlation coefficients > 0.95 on both datasets.

Performance

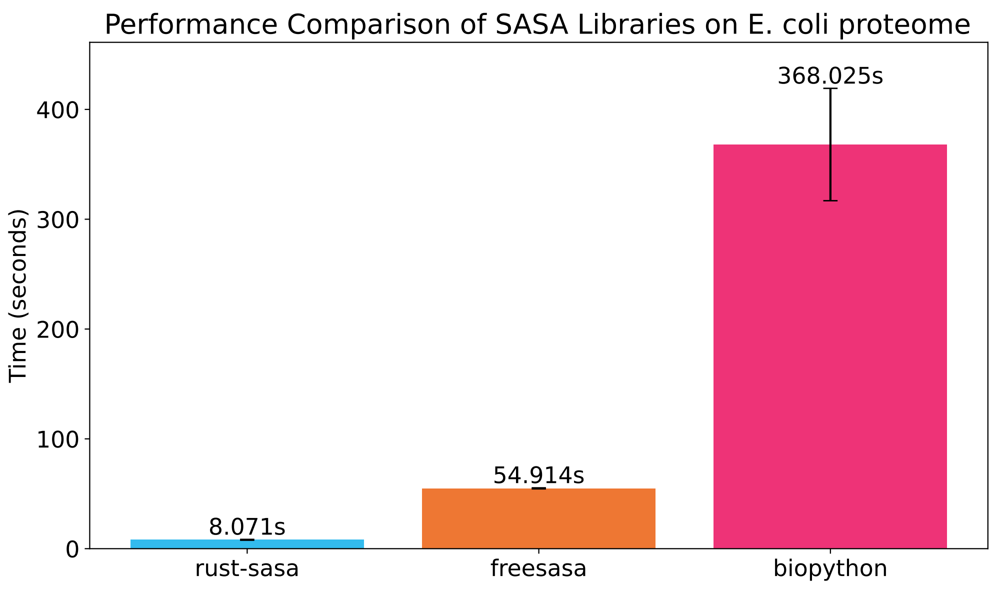


Figure 1: Comparing Freesasa, RustSasa, and Biopython performance on E. coli proteome

We evaluated the performance of Freesasa, RustSASA, and Biopython (Cock et al., 2009) on the predicted E. coli proteome using Hyperfine (Peter, 2023) with three runs and three warmup iterations on an Apple MacBook Air with an M3 processor and 24GB of unified memory. All methods utilized parallel processing across eight cores: GNU parallel (Tange, 2011) for Freesasa and Biopython, and RustSASA's internal parallelization.

RustSASA processed the entire proteome in ~8 seconds compared to ~55 seconds for Freesasa and ~368 seconds for Biopython, representing 7× and 46× speed improvements, respectively.

Conclusion

RustSASA provides a significant advancement in SASA calculation performance while maintaining accuracy, addressing a bottleneck in computational structural biology. The 7× speed improvement over current standards enables previously intractable analyses of large protein datasets and molecular dynamics simulations. By providing interfaces for multiple programming languages alongside a command-line tool, RustSASA ensures broad accessibility across the research community. As structural biology datasets continue to expand, efficient computational tools like RustSASA become essential for advancing our understanding of protein structure and function.

Acknowledgements

We would like to thank Rodrigo Honorato for his valuable contributions to this project.

References

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & others. (2009). Biopython: Freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

Mitternacht, S. (2016). FreeSASA: An open source c library for solvent accessible surface area calculations. *F1000Research*, *5*, 189. https://doi.org/10.12688/f1000research.7931.1

Peter, D. (2023). *Hyperfine*. https://github.com/sharkdp/hyperfine

Tange, O. (2011). GNU parallel - the command-line power tool*;Login: The USENIX Magazine*, *36*(1), 42–47. http://www.gnu.org/s/parallel