Novos benchmarks para aplicações de *streaming* em Rust*

Leonardo G. Faé, Dalvan Griebler

[1] Escola Politécnica, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

leonardo.fae@edu.pucrs.br,dalvan.griebler@pucrs.br

Resumo. *Rust é uma nova linguagem de programação de baixo nível com foco em desempenho e segurança. RustStreamBench é um conjunto de benchmarks criados para medir a performance de Rust em aplicações de processamento de stream. Propomos duas aplicações novas:* `sobel` *e* `latbol`*, para adicionar a esse conjunto, com características de processamento diferentes.*

1. Introdução

Rust [The Rust Project 2025] é uma linguagem de programação de baixo nível que vem ganhando destaque nos últimos anos, ao ponto de ser a única linguagem além de C a ser incluída no Kernel do Linux[1]. Rust foca em desempenho e segurança, garantindo ao programador que, se o programa compilar e não usar a palavra chave `unsafe`, ele não contém comportamento indefinido (*Undefined Behavior*).

O foco de Rust em performance justifica iniciativas de medir o seu desempenho. Uma dessas iniciativas foi o *RustStreamBench* [Pieper et al. 2021], que fornece 4 aplicações para medir o desempenho de Rust em aplicações de processamento de stream. Neste trabalho, propomos duas aplicações novas: *latbol* e *sobel*.

2. Aplicações e Resultados

A aplicação *sobel*[2] opera sobre um conjunto de imagens. Ela transforma essas imagens em tons de cinza, e após aplica um filtro sobel [Kanopoulos et al. 1988] sobre elas. *Sobel* difere do *image-processing*, outra aplicação de *RustStreamBench*, por ter dois estágios de processamento com custos bastante diferentes, com um primeiro estágio computacionalmente simples, e um segundo complexo. Além disso, *sobel* não efetua chamadas para uma biblioteca externa, como *image-processing*.

A aplicação *latbol*[3] realiza uma simulação de fluídos usando o método Lattice Boltzman [Chen and Doolen 1998]. O código também foi escrito manualmente sem chamadas à uma biblioteca externa. É composto de 4 estágios de processamento heterogêneos, com os o primeiro e último estágios sendo menos custosos que o segundo e terceiro.

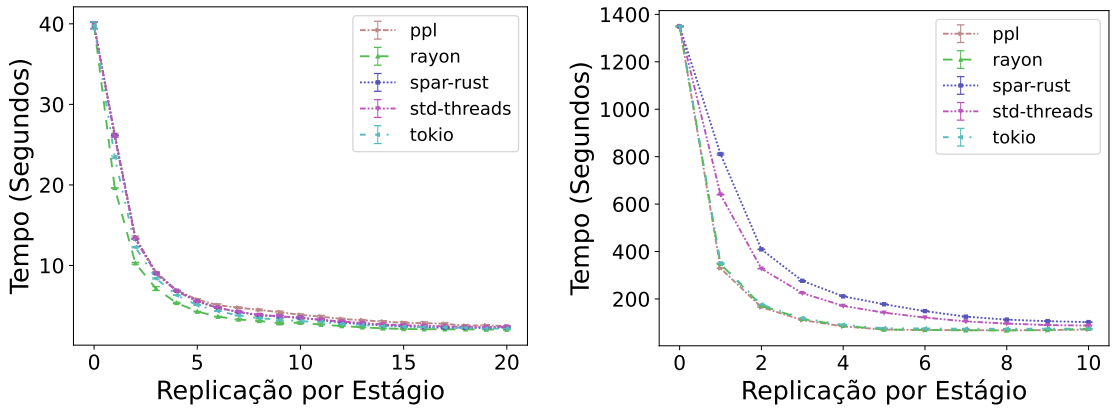
A Figura 1 mostra os resultados de executar as aplicações com uma séria de *runtimes* em Rust. Os experimentos foram executados em uma máquina com dois Intel(R) Xeon(R) Silver 4210, 144GB de RAM e 4 discos rígidos com 2TB em RAID10. As runtimes escolhidas são as mesmas de [Faé et al. 2023], com exceção do *ppl* [Besozzi 2024], que substituiu o *ssp* [Pieper et al. 2019].

*Agradecimentos à FAPERGS 09/2023 PqG (Nº 24/2551-0001400-4), e ao CNPq *Research Program* (Nº306511/2021-5) por financiarem este trabalho.

[1] *Git Pull*: `https://lore.kernel.org/lkml/202210010816.1317F2C@keescook/`

[2] código adaptado de `https://github.com/dangreco/edgy`

[3] código adaptado de `https://github.com/ndbaker1/bloe`



(a) `sobel` **(b)** `latbol`

Figura 1. Desempenho de diversas *runtimes* nas aplicações

3. Conclusões

Os resultados indicam que as *runtimes* se comportam de maneira semelhante na aplicação *sobel*. *Latbol* mostra uma diferença maior entre elas, com *spar-rust* apresentando a pior escalabilidade. O fato da maioria das *runtimes*, em ambas aplicações, serem melhor do que *std-threads* indica que as suas estruturas internadas de distribuição de carga são otimizadas para lidar com o processamento heterogêneo dessas aplicações. Como trabalhos futuros, podemos incluir mais aplicações com características de processamento diferente.

Referências

Besozzi, V. (2024). Ppl: Structured parallel programming meets rust. In *32nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing*.

Chen, S. and Doolen, G. D. (1998). Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(Volume 30, 1998):329–364.

Faé, L. G., Hoffman, R. B., and Griebler, D. (2023). Source-to-source code transformation on rust for high-level stream parallelism. In *Proceedings of the XXVII Brazilian Symposium on Programming Languages*, SBLP '23, page 41–49, New York, NY, USA. Association for Computing Machinery.

Kanopoulos, N., Vasanthavada, N., and Baker, R. L. (1988). Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2).

Pieper, R., Griebler, D., and Fernandes, L. G. (2019). Structured Stream Parallelism for Rust. In *XXIII Brazilian Symposium on Programming Languages (SBLP)*, SBLP'19, pages 54–61, Salvador, Brazil. ACM.

Pieper, R., Löff, J., Hoffmann, R. B., Griebler, D., and Fernandes, L. G. (2021). High-level and Efficient Structured Stream Parallelism for Rust on Multi-cores. *Journal of Computer Languages*, 65:101054.

The Rust Project (2025). The rust reference.