Benchmarks em Hush

João Pedro Reis Teclo de Miranda
UFMG
Belo Horizonte, Brasil

Fernando Magno Quintão Pereira
Orientador
UFMG
Belo Horizonte, Brasil

***Resumo*—Este estudo apresenta uma análise de desempenho comparando Hush, uma linguagem moderna de shell script inspirada em Lua, com Python através de uma série de microbenchmarks. A pesquisa implementa dez tarefas computacionais do Computer Language Benchmarks Game em ambas as linguagens para avaliar seu desempenho relativo. Utilizando a ferramenta de benchmark hyperfine, o estudo conduziu 50 execuções com 10 rodadas de aquecimento para cada programa em um ambiente Linux. Os resultados mostram que Hush apresenta desempenho 3-10 vezes mais lento que Python em nove dos dez benchmarks, com exceção do benchmark regex-redux, onde Hush demonstrou desempenho superior. A hipótese é que esta exceção pode ser atribuída à implementação do Hush em Rust e sua biblioteca de expressões regulares. Os resultados contribuem para a compreensão das características de desempenho do Hush em comparação com linguagens de script estabelecidas.**

I. INTRODUÇÃO

Ao se iniciar um projeto de desenvolvimento de software, uma das escolhas chave é a de uma linguagem de programação a ser usada. Tal escolha pode ser motivada por diversos fatores, como o escopo e objetivos do projeto, ecossistema de bibliotecas e frameworks, familiaridade dos programadores e considerações de performance. Esse processo de seleção pode ser guiado pelo uso de critérios objetivos de avaliação, resultando em uma melhor tomada de decisão [1].

Uma das possíveis técnicas a serem utilizadas nesse contexto é o Benchmarking, que consiste na execução de um programa ou conjunto de programas de forma a avaliar a performance de um sistema. Diante do cenário apresentado, o presente trabalho tem como objetivo produzir um conjunto de microbenchmarks da linguagem de programação Hush que serão comparados a programas equivalentes na linguagem Python.

Hush é uma linguagem de programação projetada como uma alternativa moderna às linguagens de shell script tradicionais. Inspirada por Lua, Hush oferece funcionalidades de linguagens de programação de uso geral, como estruturas de dados não-triviais e tratamento de erros [2]. O objetivo é que esse conjunto de benchmarks elaborado por este trabalho possa contribuir para o entendimento da performance de Hush e como ela se compara a linguagens de script já estabelecidas.

II. REFERENCIAL TEÓRICO

Benchmarking é uma técnica amplamente utilizada na literatura de Ciência da Computação. Em 1976 foi publicado o primeiro programa explicitamente projetado para benchmarking, o Whetstone, escrito em Algol 60 [3]. Em particular, alguns trabalhos utilizaram o Computer Language Benchmarks Game como conjunto de programas a ser analisado.

Em [4], os autores investigaram a performance da linguagem Rust, comparando-a com a linguagem C como parâmetro de referência. Um dos conjuntos de programas usados no benchmark foi o Computer Language Benchmarks Game. Os resultados do estudo indicaram que Rust foi 1.35x mais lento ao executar os programas desse benchmark, quando comparado a C. Os pesquisadores identificaram duas fontes principais de overhead: checagens em tempo de execução inseridas pelo compilador e restrições de design da linguagem. Porém, quando essas restrições foram retiradas, a performance de Rust tornou-se comparável com C, sugerindo que o overhead é principalmente uma contrapartida das garantias de segurança da Rust, em vez de uma limitação inerente.

Outro trabalho relevante no contexto [5], os autores utilizaram os programas de Computer Language Benchmarks Game para analisar a eficiência energética de 27 linguagens de programação presentes no repositório. O estudo conclui que linguagens mais rápidas nem sempre são as mais eficientes em termos de energia, com linguagens compiladas apresentando melhor desempenho, enquanto linguagens interpretadas consomem mais energia. Foi encontrada uma correlação fraca entre o uso de memória e o consumo de energia, sugerindo a influência de outros fatores. A partir disso, foi possível entender o impacto do consumo de energia ao selecionar uma linguagem de programação.

III. METODOLOGIA

O Computer Language Benchmarks Game é um projeto que compara a performance de diferentes linguagens de programação a partir de um repositório de programas *crowd-sourced* [6]. O site do projeto apresenta dez tarefas computacionais, exigindo que as implementações submetidas por seus usuários sigam estritamente os algoritmos especificados na descrição de cada problema.

Para compor o conjunto de microbenchmarks deste trabalho, foram escolhidas cinco das tarefas do Benchmarks Game e implementados programas em Hush para resolvê-las, baseado nas descrições dos algoritmos fornecidos na página de cada tarefa, e nas implementações crowd-sourced em outras linguagens de programação. Em relação aos programas Python, esses foram coletados do próprio repositório de programas do Benchmarks Game, dentre os vários disponíveis para cada tarefa foram escolhidos os programas mais rápidos desde que não usassem funcionalidades indisponíveis em Hush, como concorrência e paralelismo, por exemplo.

Foi utilizado o interpretador de Hush versão 0.1.4 e o interpretador Python versão 3.12.8. Os benchmarks foram executados usando a ferramenta hyperfine versão 1.19.0 [7], configurada para realizar 50 execuções de cada programa e 10 execuções de warmup, para minimizar efeitos de aleatoriedade. Todos os testes foram executados em uma máquina Linux com processador Intel Core i7-2670QM e 8GB de RAM. O código produzido para este projeto está disponível em um repositório do GitHub: https://github.com/joaoxyz/hush-benchmarks.

IV. RESULTADOS

Seguindo os passos relatados no capítulo de metodologia foram executados os benchmarks medindo a performance de execução das implementações em Hush e Python das tarefas

do Benchmarks Game. A Figura 1 mostra os resultados da comparação.

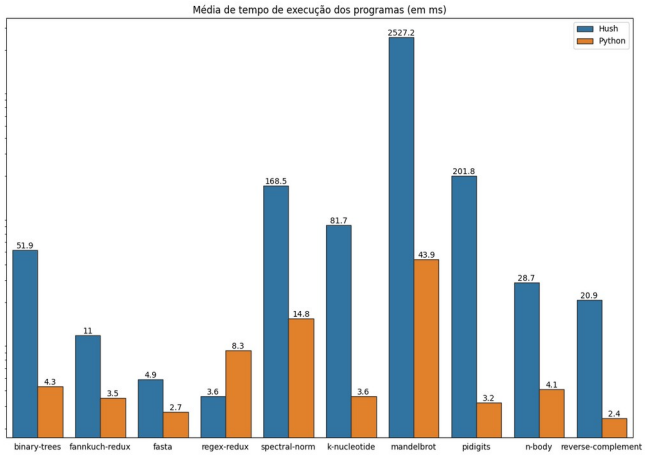

Fig. 1. Tempos médios de execução dos benchmarks (eixo y em escala logarítmica)

Em nove dos dez benchmarks, a implementação em Python demonstrou superioridade em relação à de Hush, registrando tempos de execução entre 3 a 10 vezes inferiores. Era esperado que houvesse essa diferença tendo em vista a maturidade do ecossistema Python, que tem décadas de otimização e desenvolvimento em seu interpretador padrão. Considerando a sua relativa juventude, é plausível que o interpretador de Hush ainda não disponha do mesmo grau de otimização para este tipo de carga de trabalho.

O único programa em que a performance de Hush venceu Python foi o regex-redux. Esse programa consiste na manipulação dos dados de entrada usando expressões regulares. Tal resultado constitui um indício de que a biblioteca de expressões regulares de Rust, linguagem na qual o interpretador de Hush é implementado, detém um desempenho consideravelmente superior ao do módulo re padrão do Python para as operações específicas deste benchmark.

V. Limitações e trabalhos futuros

A presente investigação limitou-se a um conjunto específico de microbenchmarks focados na utilização da CPU. Para uma avaliação exaustiva, seria pertinente incluir na análise o consumo de memória, o desempenho em tarefas de entrada/saída (I/O), uma comparação com outras linguagens de shell ou comumente utilizadas para scripting, como Bash e Lua e a utilização de aplicações reais no benchmarking.

References

[1] *NAIDITCH, D. (n.d.). Selecting a programming language for your project. 17th DASC. AIAA/IEEE/SAE. Digital Avionics Systems Conference. Proceedings (Cat. No.98CH36267), 1998. doi:10.1109/dasc.1998.741494.*

[2] BASTOS, G. Hush: A lua-based shell language. UFMG, 2021.

[3] *WEICKER, R. An overview of common benchmarks. Computer, v. 23, n. 12, p. 65–75, 1990.*

[4] *ZHANG, Y.; ZHANG, Y.; PORTOKALIDIS, G.; XU, J. Towards understanding the runtime performance of Rust. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. ACM, 2022. (ASE '22). http://dx.doi.org/10.1145/3551349.3559494.*

[5] *PEREIRA, R.; COUTO, M.; RIBEIRO, F.; RUA, R.; CUNHA, J.; FERNANDES, J. P.; SARAIVA, J. Energy efficiency across programming languages: how do energy, time, and memory relate? In: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering. ACM, 2017. (SPLASH '17). http://dx.doi.org/10.1145/3136014.3136031.*

[6] GOUY, I. The Computer Language Benchmarks Game. 2024. https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html.

[7] PETER, D. hyperfine - A command-line benchmarking tool. https://github.com/sharkdp/hyperfine.