Avaliação de desempenho, custo e eficiência energética de linguagens de programação

Discente: Michel Tavares de Oliveira
Orientador: Jean Carlos Teixeira de Araújo

Universidade Federal do Agreste de Pernambuco

Maio de 2024

Motivação

- A indústria de TI corresponde por 3% a 4% das emissões globais, cerca do dobro das emissões geradas pela aviação civil
- Recursos finitos de energia e o aumento das tarifas energéticas.
- O impacto do consumo excessivo de energia nos custos operacionais, no preço dos serviços para o usuário final e no meio ambiente.
- Ter um sistema eficiente energeticamente é crucial para manter a viabilidade de serviços computacionais.

Questões de pesquisa

- Questão de pesquisa 1. Qual linguagem de programação é mais eficiente energeticamente?
- Questão de pesquisa 2. A linguagem mais rápida, é mais eficiente energeticamente?
- Questão de pesquisa 3. A linguagem que possui menor potência em Watts, foi a mais eficiente?

Trabalhos ralacionados

- Energy efficiency across programming Languages: How Do Energy, Time, and Memory Relate?
- Towards a green ranking for programming languages
- Analyzing programming languages's energy consumption: an empirical study

Avaliação de desempenho de sistemas

- Objetivo de entender como um sistema se comporta em termos de um determinada métrica com uma carga de trabalho;
- Auxiliar na tomada de decisão como investimentos em hardware e ou software;
- Otimização e melhorias a partir da análise dos resultados.

Green software

- Abordagem para o desenvolvimento e uso do software de maneira mais sustentável;
- Objetivo de reduzir impactos ambientais;
- Estratégias que buscam minimizar o consumo de recursos naturais.

Eficiência energética

- Utilização da energia de forma consciente e eficaz.
- Objetivo de minimizar desperdícios
- Otimizar o uso de energia de forma que a energia consumida seja proporcional ao trabalho realizado.

Joule

- É a unidade de energia no Sistema Internacional de Unidades, utilizada para medir energia mecânica ou térmica;
- Na energia mecânica, 1 Joule equivale a energia necessária para aplicar força 1 Newton por 1 metro;
- Na energia térmica, 1 Joule equivale a energia necessária para aumentar a temperatura da água a 1 grau.

Watt

- É a unidade de potência no Sistema Internacional de Unidades, sendo a potência media da quantidade de energia em um determinado tempo;
- 1 Watt equivale a 1 Joule por segundo, logo, um dispositivo que consome 1 Watt está consumindo um Joule por segundo.

$$P = \frac{E}{t} \tag{1}$$

Quilowatt

- É uma unidade de potência referente a 1000 Watts;
- Utilizada para medir potência elétrica em aplicações residenciais, comerciais e industriais;

$$\text{Potência em quilowatts (kW)} = \frac{\text{Energia em kilojoules (kJ)}}{\text{Tempo em horas (h)}} \tag{2}$$

Quilowatt-hora

- Referente a energia produzida ou consumida no período de 1 hora.

$$\text{Energia total (kWh)} = \text{Potência em Quilowatts (kW)} \times \text{Tempo total em horas (h)} \quad (3)$$

Intel RAPL

- Otimizar o gerenciamento energético dos processadores Intel
- Monitoramento de alguns parametros como temperatura, potência e consumo energético
- Foi implementado a nível de hardware a partir da 6ª geração dos processadores Intel
- A precisão do RAPL é bastante promisora e os valores reportados são precisos suficientemente para prever e modelar sistemas [1]

RAPL Power Domain

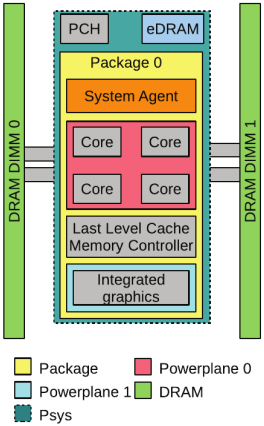


Figure 1: Intel RAPL Power Domains. [1]

Contadores de energia e limitações

- MSR_RAPL_POWER_UNIT de 32 bits sem sinal (0 a 4,294,967,295);
- O registrador começa a contar a partir da inicialização do computador;
- Quando este valor limite é atingido, o valor do registrador é zerado;
- É de grande importância considerar a reinicialização dos registradores para não obter dados incorretos durante os experimentos.

Power Capping Framework

- Ferramenta integrada ao Kernel Linux
- Permite expor informações de energia via *sysfs*;
- O framework cria de forma automática, uma árvore de diretórios com diversos objetos referente a interface de energia utilizada [3].

Árvore de diretórios Power Capping Framework

```
/sys/devices/virtual/powercap
└── intel-rapl
    ├── enabled
    ├── intel-rapl:0
    │   ├── constraint_0_max_power_uw
    │   ├── constraint_0_name
    │   ├── constraint_0_power_limit_uw
    │   ├── constraint_0_time_window_us
    │   ├── constraint_1_max_power_uw
    │   ├── constraint_1_name
    │   ├── constraint_1_power_limit_uw
    │   ├── constraint_1_time_window_us
    │   ├── device -> ../../intel-rapl
    │   ├── enabled
    │   ├── energy_uj
```

Figure 2: Árvore de Diretórios Power Capping Framework

Shell

- Interface entre o usuário e o sistema operacional
- O Shell é uma ferramenta essencial quando o foco é ter mais controle sobre o sistema operacional [2]
- Funciona como um intermediário entre usuário e SO
- Essa interação pode ocorrer de forma iterativa e não iterativa

Shell script

- É uma linguagem de script voltada para automatização de tarefas em sistemas operacionais, sendo ela interpretada por um interpretador Shell;
- Permite realizar diversas tarefas executando apenas um arquivo de script;
- O interpretador analisa linha por linha e executa os comandos encontrados de forma sequencial;
- Bastante útil ao executar diversos comandos, assim como a possibilidade de realização de tarefas repetitivas e automáticas.

Bash

- Bash é um shell desenvolvido por Brian Fox no Projeto GNU
- Atualmente é o Shell padrão de diversas distribuições Linux, como Ubuntu, Debian e Manjaro.

GNU Time

- Utilizada para medir tempo e recursos consumidos por uma aplicação durante sua execução;
- Utilização é bastante simples, permitindo que os usuários escolham os dados específicos que desejam extrair e analisar.

```
/usr/bin/time ./meu_programa > output.txt
```

The Computer Language Benchmark Game

- É um projeto de software livre que fornece um repositório com uma variedade de problemas simples que podem ser implementados em diversas linguagens de programação;
- Um web site que centraliza dados sobre códigos fontes, execução de teste e resultados;
- O projeto fornece, além do código fonte, informações sobre compilação e execução dos algorítimos.

CLBG Benchmarks

Benchmarks	Descrição
fannkuch-redux	Acesso indexado a minúsculas sequências inteiras
n-body	Dupla precisão para cálculo de N-body
spectral-norm	Autovalor usando o método da potência
pidigits	Streaming de aritmética de precisão arbitrária
regex-redux	Combina DNA e substitui por padrões mágicos
fasta	Gerar e escrever sequências aleatórias de DNA
k-nucleotide	Atualiza hashtable e sequências de k-nucleotídeos
reverse-complement	Complemento reverso de sequências de DNA
binary-trees	Aloca e desaloca muitas árvores binárias
mandelbrot	Gera um conjunto de Mandelbrot em arquivo de bitmap portátil

Linguagens de programação

Linguagem	Versão	Compilador Open Source (Ubuntu 22.04)
Ada	10.5.1	GNAT GPL Compiler
C	11.4.0	GCC
C#	7.0.115	Mono
C++	11.4.0	GCC
Chapel	1.29.0	Chapel Compiler
Dart	3.2.6	Dart SDK
Erlang	26.2.2	Erlang OTP
F#	7.0.115	F# Compiler
Fortran	11.4.0	GFortran
Go	1.18.1	Go Compiler
Haskell	8.8.4	GHC Haskell Compiler
Java	19.0.2	OpenJDK
Javascript	18.19.0	V8
Julia	1.9.3	Julia Compiler
Lua	5.3.0	LuaJIT
Ocaml	4.13.1	OCaml Compiler
Perl	5.34.1	Perl Compiler
Php	8.2.15	PHP Compiler
Python	3.10.12	Python Interpreter
Racket	8.2.0	Racket Compiler
Ruby	3.0.2	Ruby Compiler
Rust	1.75.0	Rustc Compiler
Swift	5.9.0	Swift Compiler

Hardware

Especificação	Descrição
Modelo	Intel® Core™ i5-10210U
Geração	10ª Geração
Codinome	Comet Lake
Nº de núcleos	4
Threads	8
Frequência base	1.60 GHz
Frequência turbo max	4.20 GHz
Cache	6 MB Intel® Smart Cache
Litografia	14 nm
TDP	15 W

Hardware

Especificação	Descrição
Memória DRAM	8GB (Dual Channel)
Frequência da Memória DRAM	2666MHz
Tipo de Memória DRAM	DDR4
Armazenamento Interno	256GB
Tipo do Armazenamento Interno	PCIe NVMe M.2
Potência Máxima da Fonte de Alimentação	45W

Sistema operacional

Especificação	Descrição
Distribuição	Ubuntu
Versão	22.04 LTS
Codinome	Jammy Jellyfish
Ambiente de desktop	KDE Plasma
Kernel	Linux 5.15
Arquitetura	x86 64 bits
Tipo de instalação	Desktop

Scripts de instalação

- Objetivo de agilizar a instalação de pacotes e dependencias das linguagens de programação;
- Garante que todas as dependencias serão instaladas para execução dos experimentos.

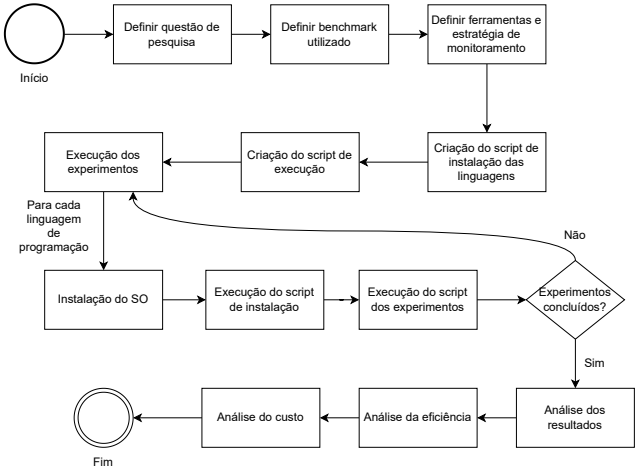
Scripts de execução

- Objetivo coletar os dados reportados pelo RAPL salva em um arquivo para uma análise posterior;
- 100 execuçãoe cada problema do CLBG;
- intervalo de 10 minutos entre os problemas;
- Considerado a média da 100 execuções para calcular os resultados.

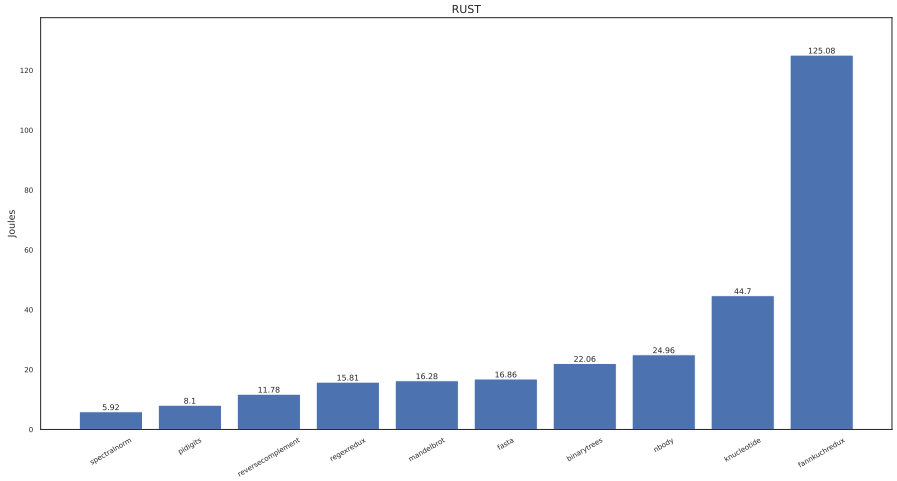
Scripts de execução

```
Algoritmo lerEnergia

Variáveis:
    energia_psys_inicio
    energia_psys_fim
    energia_psys_total
    energia_dram_inicio
    energia_dram_fim
    energia_dram_total
    total
Inicio:
    i ← 1
    Repita:
        Leia(energia_psys_inicio)
        Leia(energia_dram_inicio)

        Execute(<comando do algorítmo>)

        Leia(energia_psys_fim)
        Leia(energia_dram_fim)
        Leia(temperatura_da_cpu)

        psys_total ← (psys_fim - psy_inicio)
        dram_total ← (dram_fim - dram_inicio)

        total ← (cpu_total + dram_total)

    até i <= 100
Fim
```

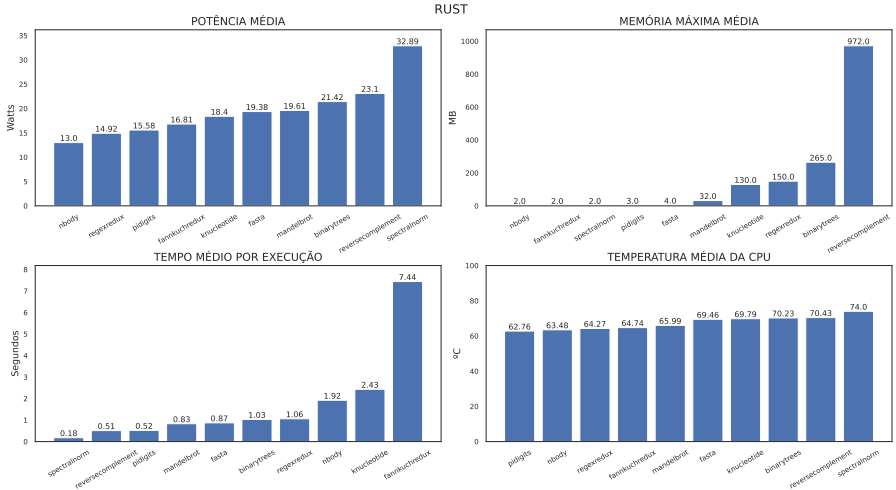
Designin de excecução



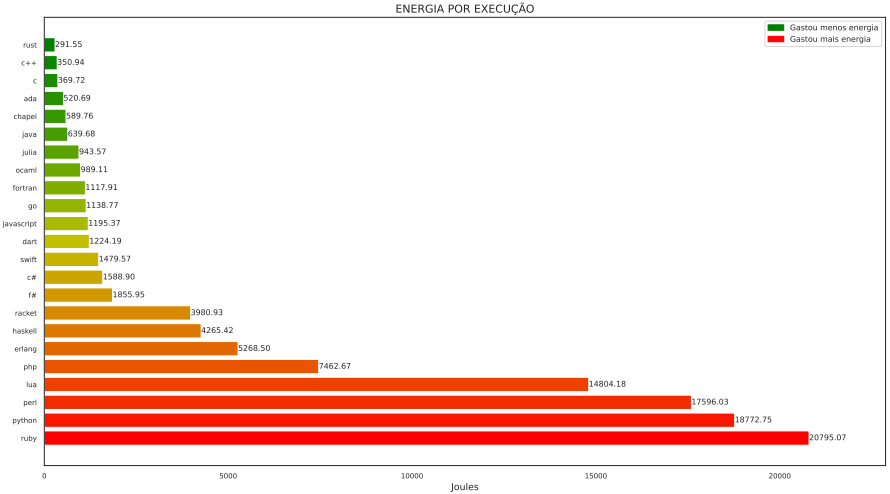
Consumo em Joules por problema



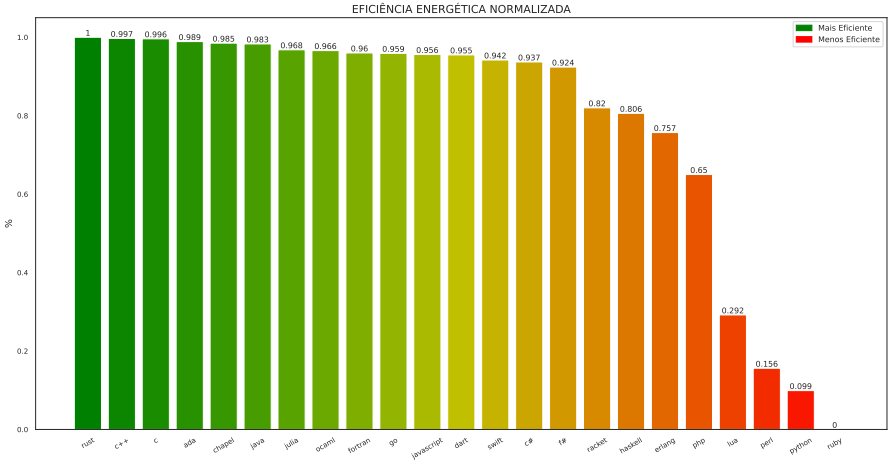
Potência, memória, tempo e temperatura



Energia por execução



Eficiência energética



Análise do custo financeiro por 1000 execuções

- A tarifa média considerada foi de 0.73 centavos por kWh
- Utilizamos o consumo total em Joules e convertemos para kWh
- Multiplicamos o consumo total em kWh por 1000 execuções

$$\text{Consumo em kWh} = \frac{\text{Consumo em Joules}}{3.600.000} \tag{4}$$

Linguagens de programação

Linguagem	Joules	kWh	kWh por mil execuções	Custo por mil execuções (R$)
Rust	291,55	0,000081	0,081	0,05913
C++	350,94	0,000098	0,098	0,07154
C	369,72	0,000103	0,103	0,07519
Ada	520,69	0,000145	0,145	0,10585
Chapel	589,76	0,000164	0,164	0,11972
Java	639,68	0,000178	0,178	0,12994
Julia	943,57	0,000262	0,262	0,19126
Ocaml	989,11	0,000275	0,275	0,20075
Fortran	1117,91	0,000310	0,310	0,2263
Go	1138,77	0,000316	0,316	0,23068
Javascript	1195,37	0,000332	0,332	0,24236
Dart	1224,19	0,000340	0,340	0,2482
Swift	1479,57	0,000411	0,411	0,30003
C#	1588,90	0,000441	0,441	0,32193
F#	1855,95	0,000516	0,516	0,37668
Racket	3980,93	0,001106	1,106	0,80738
Haskell	4265,42	0,001184	1,184	0,86432
Erlang	5268,50	0,001463	1,463	1,06799
PHP	7462,67	0,002073	2,073	1,51329
Lua	14804,18	0,004112	4,112	3,00176
Perl	17596,03	0,004888	4,888	3,56824
Python	18772,75	0,005214	5,214	3,80622
Ruby	20795,07	0,005777	5,777	4,21721

Qual linguagem de programação é mais eficiente energeticamente?

- Rust (291.55 Joules)
- C++ (350.94 Joules)
- C (369.72 Joules)

A linguagem mais rápida, é mais eficiente energeticamente?

- Não;
- Julia aoresentou maior tempo de execução em relação a Go e Chapel, mas conseguiu apresentar melhor eficiência, mesmo sendo mais lenta.

A linguagem que apresentou menor potência em watts, foi a mais eficiente?

- Não;
- Pontência menor resulta em um maior tempo de execução;
- Linguagens com maior potência necessitavam de um menor tempo de execução.

[1] Kashif Nizam Khan et al. "RAPL in Action: Experiences in Using RAPL for Power Measurements". In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3.2 (2018). ISSN: 2376-3639. DOI: 10.1145/3177754. URL: https://doi.org/10.1145/3177754.

[2] Eric S Raymond. *The art of Unix programming*. Addison-Wesley Professional, 2003.

[3] The Linux Kernel Archives. *Power Capping Framework Documentation*. Acessado em 4 de fevereiro de 2024. 2024. URL: https://www.kernel.org/doc/html/next/power/powercap/powercap.html.

EOF