UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS APUCARANA*

Lissa Guirau Kawasaki

RELATÓRIO TÉCNICO

Arquitetura e Organização de Computadores

APUCARANA – PR

2024

Lissa Guirau Kawasaki

RELATÓRIO TÉCNICO

Arquitetura e Organização de Computadores

Trabalho apresentado à disciplina de Arquitetura e Organização de Computadores do curso de Bacharelado em Ciência da Computação.

Professor: Guilherme Henrique de Souza Nakahata.

APUCARANA – PR

2024

SUMÁRIO

INTRODUÇÃO 04

CAPÍTULO 1: OBJETIVOS 05

CAPÍTULO 2: MOTIVAÇÃO E RECURSOS UTILIZADOS 05

2.1 Motivação 06

2.3 Linguagem de programação e demais informações 07

2.2 Estrutura de Dados 07

CAPÍTULO 3: RESULTADOS 09

CONCLUSÃO 11

REFERÊNCIAS 12

INTRODUÇÃO

A disciplina Arquitetura e Organização de Computadores abrange diferentes áreas da computação e mecânica, estando presente em cursos técnicos, engenharias e bacharelados. Trata-se de uma disciplina fundamental para a formação de profissionais em tais áreas, isto é, garante que os conceitos estudados em algoritmos, programação e outras matérias teóricas, como variáveis, vetores e ponteiros tenham seu nível de abstração reduzido e suas instruções sejam traduzidas em ações físicas, como armazenamento e manipulação de dados e interação de componentes, e sejam mais bem compreendidos ao estudarmos a arquitetura e funcionamento de sua principal plataforma. Através das aulas ministradas, a compreensão da matéria permite a compreensão da linguagem da máquina, uma habilidade crucial para o desenvolvimento de softwares eficientes, especialmente em áreas relacionadas a computação de alto desempenho.

Para que o ensino de conceitos de arquitetura e organização de computadores seja concluído, é essencial considerar o desempenho de um processador como parâmetro na avaliação de máquinas. Nessa análise, consideramos não apenas a velocidade do processador, mas a velocidade do conjunto de instruções, relacionado a escolha de linguagem de implementação utilizada no programa, a eficiência da plataforma e os resultados ao executar o programa desejado. As medidas tradicionais de avaliação de um processador e desempenho computacional são feitas pelo processo de “benchmarking”, que, através de uma seleção estratégica de critérios avaliativos, proporciona uma melhor visualização de sistemas e suas utilidades e garante uma análise precisa e completa do comportamento de um programa. Atualmente, a Standard Performance Evaluation Corporation (SPEC), um consórcio sem fins lucrativos, é quem estabelece e define o conjunto de pacotes de benchmark mais conhecidos pela indústria de tecnologia, direcionando as avaliações de desempenho de sistemas computacionais existentes, e garantindo a padronização de performance para futuras novas gerações de sistemas de computação.

OBJETIVOS

O tema conecta conceitos teóricos a suas principais aplicações práticas, dessa forma, dentro da disciplina, cria-se a oportunidade de colocar em prática os aprendizados de linguagens de programação, para uma aplicação de criação de benchmarking de forma em que, é colocado em prática as principais métricas de referência e testes definidas pela SPEC, nomeadamente, métricas de base, pico, velocidade e taxa. No desenvolvimento, é necessário ponderar o desenvolvimento de testes eficazes, que explorem os recursos e limites do sistema de forma eficiente e precisa, e utilizando o material disponibilizado pela disciplina, interpretar e realizar análises dos resultados dos testes, identificando questões de desempenho e melhores oportunidades de otimização.

O objetivo principal do código, como citado anteriormente, é interpretar e analisar a performance de funcionamento de um programa de forma prática. Em tese, utilizaremos de um programa que interpreta operações aritméticas e, a partir de bibliotecas disponíveis nas linguagens de programação, realizaremos a simulação de um benchmark, utilizando parâmetros de tempo de geração, uso de memória, uso da Unidade Central de Processamento e uso de disco e, por análises críticas, garantimos uma melhor execução em relação a performance do sistema.

MOTIVAÇÃO E RECURSOS UTILIZADOS

2.1 Motivação.

A principal motivação para a criação de um benchmark é a busca pelo melhor desempenho possível em um programa independente de sua plataforma. Desde computadores pessoais ou servidores corporativos, para profissionais e desenvolvedores é evidente a necessidade de otimizar o uso dos recursos computacionais para garantir melhor funcionamento em relação a eficiência, velocidade e confiabilidade. Com o avanço constante dos meios tecnológicos, a necessidade de comparação entre sistemas e softwares se torna crucial para o mercado, para isso, é necessário de uma plataforma transparente que realize as comparações e permita que usuários e empresas tomem as melhores decisões em relação às suas necessidades. Essa comparação transparente impulsiona a inovação no mercado de tecnologia e promove uma concorrência acirrada.

A flexibilidade encontrada nos pacotes de benchmarks torna-os ferramentas indispensáveis para a avaliação de sistemas e softwares, isto é, torna-se essencial para discentes e profissionais na área de ciência da computação e engenharia de software. Para um profissional, a habilidade de identificação de ineficiências e problemas é necessária para um melhor gerenciamento e aplicação de códigos.

Através da implementação do código fonte feito, é possível, de forma menos subjetiva, compreender as principais motivações e objetivos acerca das informações citadas anteriormente, dessa forma, faremos uma análise da estrutura de dados para melhor visualização de suas funções e ideias.

2.2 Linguagem de Programação

Para o código, foi utilizado da linguagem de programação Python, uma linguagem que disponibiliza ferramentas eficientes e versáteis para o trabalho, a linguagem oferece uma gama de benefícios que impulsionam a produtividade e a resolução de problemas, sendo assim, a sua sintaxe clara e intuitiva e sua abundância de ferramentas torna ideal para a utilização em um programa que tem como seu principal objetivo a solução de problemas.

2.3 Estrutura de Dados.

De maneira geral, o código fonte funciona a partir de uma estrutura de cálculos aritméticos e estatísticos. As bibliotecas utilizadas, *random, statistics, time* e *psutil* garantem o funcionamento do código a partir de suas funções; a biblioteca random é utilizada para gerar números aleatórios, e a biblioteca statistics disponibiliza de atalhos para o cálculo de funções aritméticas utilizadas no código, garantindo otimização e praticidade. As bibliotecas time e psutil são essenciais para a medição de performance do programa, isto é, dando objetivo ao benchmark.

A princípio, é criada uma função “listaAleatoria”, e, como seu título indica, ela exerce a geração de listas aleatórias – Essas serão consideradas o núcleo do código, afinal, a partir dessa função será possível compreender as métricas. – A função utiliza de um *loop* para a criação de um número definido de listas, com valores entre 0 e 100. A próxima função “calcularEstatisticas” cria variáveis onde serão armazenados e calculados os valores aritméticos a serem medidos pelo benchmark.

```python
def calcularEstatisticas(lista, i):
    total = 0
    Maior = lista[0]
    Mnor = lista[0]
    for num in lista:
        Total += num
        if num > Maior:
            Maior = num
        if num < Menor:
            Menor = num
    Média = total / len(lista)

    Moda = max(set(lista), key=lista.count)
    lista.sort()
    n = len(lista)
    if n % 2 == 0:
        Mediana = (lista[n//2 - 1] + lista[n//2]) / 2
    else:
        Mediana = lista[n//2]

    soma_diferencas_quadradas = sum((x - Média) ** 2 for x in lista)
    DesvioPadrão = (soma_diferencas_quadradas / len(lista)) ** 0.5
```

Figura 1, exemplificando o código antes da utilização da biblioteca statistics.

```python
def calcularEstatisticas(lista, i):

  Média = statistics.mean(lista) # Calcula a media da lista
  Moda = statistics.mode(lista) # Calcula a moda da lista
  Mediana = statistics.median(lista) # Calcula a mediana da lista
  DesvioPadrão = statistics.stdev(lista) # Calcula o desvio padrão da lista
```

Figura 2, utilizando a biblioteca statistics e garantindo otimização do código fonte.

A função “Benchmark”, assim como o seu nome indica, é a função responsável pela métrica de performance das funções citadas, utilizando-se de bibliotecas *psutil* e *time*. A biblioteca psutil do Python se destaca como uma ferramenta importante para monitoramento e gerenciamento de desempenho do sistema, afinal, sua interface fornece acesso a concepção de benchmark de forma simples e intuitiva, sendo a escolha ideal para desenvolvedores e profissionais que desejam estudar de forma aprofundada os processos relacionados ao uso de CPU, memória, disco e rede, e otimizar o desempenho de seus sistemas.

Finalizamos o código utilizando as funções previamente citadas, e, temos que a saída resultante se concluiu com sucesso.

RESULTADO

Diante aos processos previamente realizados, o pleno funcionamento do código é atingido de forma esperada – a aplicação funcional de funções aritméticas e a medição de performance e parâmetros foi realizada. A imagem a seguir exemplifica um resultado do programa:

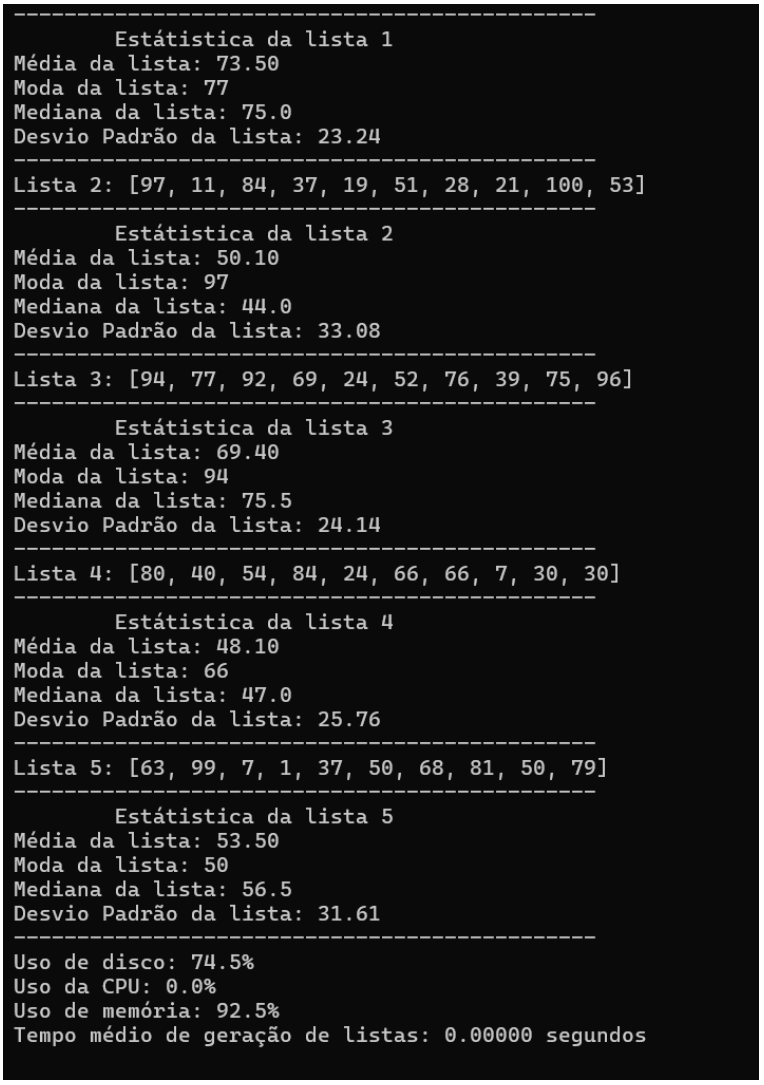

Figura 1, Acer Nitro Intel Core i5, 8GB

Para melhor visualização, o teste foi realizado em diferentes máquinas e sistemas operacionais.

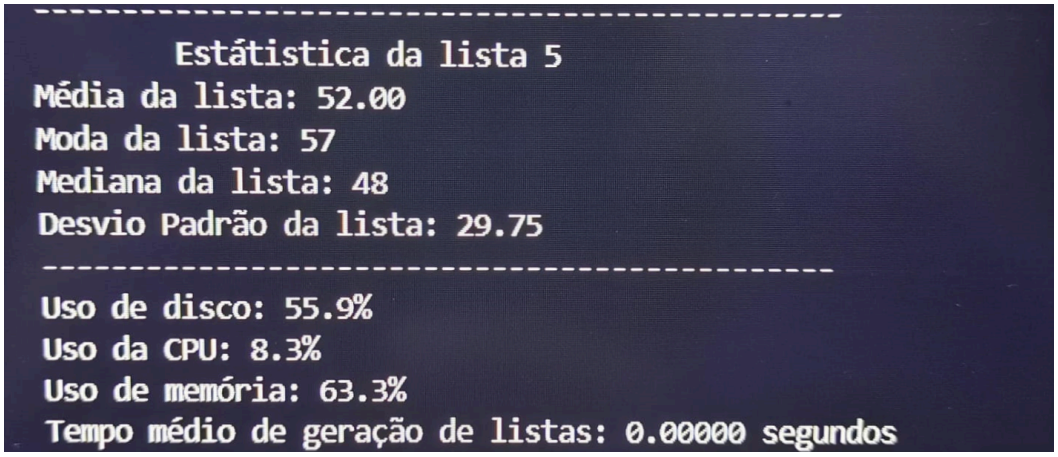

Figura 2, Acer Nitro AMD Ryzen 5, 16GB, teste realizado no sistema operacional Windows.

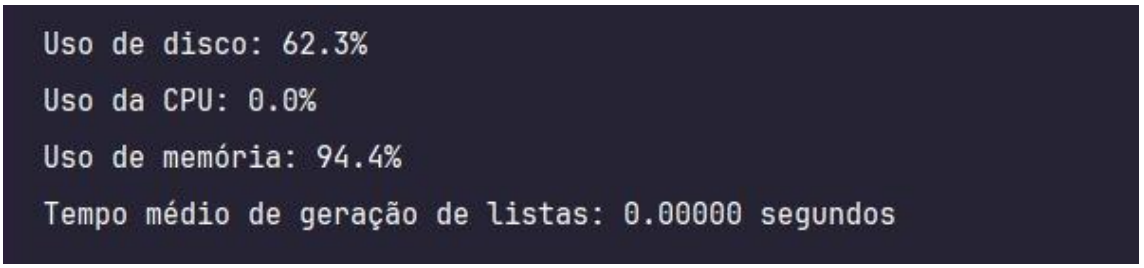

Figura 3, teste realizado no sistema operacional Windows, utilizando a IDE Visual Studio Code.

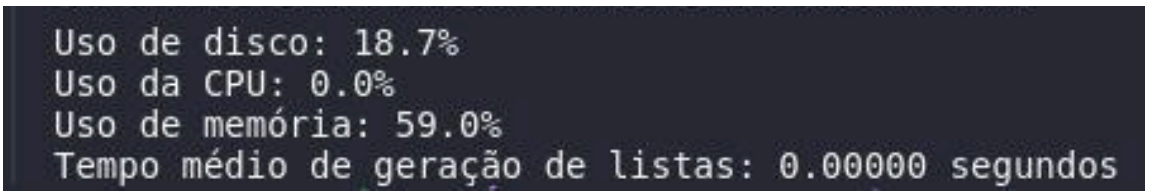

Figura 4, teste realizado no sistema operacional Linux.

CONCLUSÃO

A partir dos testes realizado acima, tomamos as seguintes conclusões:

Atualmente, computadores pessoais têm capacidade computacional para realizar performances de excelentes resultados acerca de programas de maior simplicidade, visto que, o tempo médio de processamento é mínimo, de forma que, a execução do código torna-se praticamente imediata. É possível concluir que, à medida que a complexidade de um programa aumenta, é necessário maior uso de processos em um computador, e o seu tempo tende a aumentar. Para desenvolvedores, a importância dos benchmarks é explícita, garantindo para usuários casuais uma performance rápida que garante o sentimento de “imediato” acerca dos processos.

Ao longo do trabalho, foi possível explorar a prática de aplicação de benchmarks, ultrapassando a barreira de conceitos abstratos e concluindo a essencialidade das ferramentas de desempenho para os componentes de hardware, permitindo pura compreensão da conexão entre benchmarks e as boas práticas como desenvolvedor, em Arquitetura e Organização de Computadores.

REFERÊNCIAS

STALLINGS, W. Arquitetura e organização de computadores. São Paulo (Sp): Pearson, 2010.

MORANDI, D. Um Processador Básico para o Ensino de Conceitos de Arquitetura e Organização de Computadores. Itajaí (Sp): Julho de 2008.