UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

Guilherme Fortunato Da Silva

RELATÓRIO TÉCNICO - AOC

APUCARANA – PR
2024

Guilherme Fortunato Da Silva

RELATÓRIO TÉCNICO – AOC

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

Professor: Guilherme Henrique de Souza Nakahata;

APUCARANA – PR
2024

SUMÁRIO

INTRODUÇÃO

O estudo da Arquitetura e Organização de Computadores é fundamental para a Ciência da Computação, pois proporciona uma compreensão profunda dos componentes essenciais dos sistemas computacionais e dos princípios subjacentes ao seu funcionamento,assim capacitando a projetar, desenvolver e otimizar sistemas computacionais eficientes e robustos.

Neste trabalho, irei criar um benchmark que em resumo são projetados para testar várias características do hardware ou software, como velocidade de processamento, capacidade de armazenamento, eficiência energética, desempenho gráfico, entre outros. Eles podem ser usados para comparar diferentes componentes de hardware, como CPUs, GPUs, discos rígidos, ou para avaliar diferentes sistemas

completos.

CAPÍTULO 1

OBJETIVOS

Neste trabalho em questão, o objetivo principal é criar um benchmark com a capacidade para comparar o desempenho e os requisitos de memória de diferentes métodos de preenchimento de matriz de tamanhos variados. Como não queria nada muito grande coloquei uma de 200x200 para deixar mais específico, se tornando numa matriz de 40.000 posições,no meu caso usei dois metódos, o sequencial e o aleatório.O metódo sequencial será colocado valores de 1 a 40.000 em seus respectivos lugares na matriz, enquanto o aleatório colocará esses mesmos números em posições aleatórias. O resultado mostrará o tempo de execução de cada operação além da quantidade de memória usada.

CAPÍTULO 2
MOTIVAÇÃO E RECURSOS UTILIZADOS

Baseando-se no que foi descrito anteriormente, devemos explicitar os motivos para a realização do trabalho e seu objetivo final, além dos recursos utilizados para que o trabalho seja executado de maneira eficiente e concisa.

2.1 Motivação

Como citado no capítulo 1 que trata dos objetivos do projeto em questão, minha motivação para criar um algoritmo que tem como principal operação a matriz foi observar o funcionamento das mesmas do 1° ano do curso assim imaginando o custo computacional da operação, a matéria acabou coincindindo com essa curiosidade me fazendo sentir a necessidade de avaliar e comparar o desempenho de algoritmos que envolvem operações em matrizes.

A principal motivação para a escolha do uso da linguagem 'Java' foi principalmente a necessidade de ampliar o conhecimento sobre a mesma.

Um benchmark de matriz pode ser usado para verificar se uma implementação atende aos requisitos de desempenho estabelecidos e identificar possíveis áreas de melhoria.

Assim, faz-se necessário, com as informações pertinentes acerca dos objetivos e motivações, analisar detalhadamente àqueles dados relevantes à Arquitetura e Organização Computacional, Linguagem de Programação e demais questões acerca da implementação do código em questão

2.2 Estrutura de Dados

Sob uma perspectiva geral, o código fonte contém métodos para criar, preencher e medir o tempo necessário para operações em matrizes. A estrutura do código é organizada de forma modular, com métodos separados para diferentes funcionalidades. A classe começa importando as bibliotecas . Em seguida, a declarei os métodos estáticos "criarMatriz", "sequencial", "aleatória", "medir Tempo" e "liberar Matriz". O método "criarMatriz" é responsável por criar uma matriz quadrada com o tamanho especificado que nesse caso foi de 200x200 retornado a matriz numa variável de valor inteiro; O método "sequencial" é utilizado para preencher a matriz com tamanho já especificado de 1 até 40.000 nas respectivas posições. Já o método "aleatória" eu usei para preencher os mesmos valores de 1 a 40.000 de forma que suas posições sejam escolhidas de forma aleatória pela função usada no programa, para garantir números aleatórios incrementei uma semente de aleatoriedade para garantir que a semente

seja diferente a cada execução do programa, evitando assim que a sequência de números aleatórios seja a mesma em diferentes execuções, após isso criei uma variável "posição" para encontrar as posições que não estão sendo usadas com um loop "do while", caso a variável tenha um valor booleano "false" ela será preenchida e terá seu valor alterado para "true", o programa repetirá esse loop até que todas posições sejam preenchidas. O método "medirTempo" calcula o tempo necessário para executar os preenchimentos utilizados, pegando o horário inicial em milissegundos do computador no momento que a operação foi inicializada, após isso ele faz o mesmo com o horário final, após isso ele retorna a variável "inicio"(horário inicial) subtraída pela variável "fim"(horário final) achando o tempo que a operação levou em milissegundos método "liberarMatriz" libera a memória associada à matriz para deixar os endereços limpos para próxima execução, percorrendo linha por linha e aplicando o valor 0 nos seus respectivos elementos, tornando assim mais confiável. No método principal ou "main" como descrito no código, o programa primeiro 'chama' o método "criarMatriz" que como dito anteriormente cria a matriz, após isso ele aplica o método "medirTempo" no método "sequencial" assim pegando o tempo necessário para operação e mostra o mesmo para o usuário, logo após o programa cria uma variável "memóriaSequencial" que basicamente consiste em uma operação que subtraí a memória total pela anteriormente usada ,mostrando em seguida a memória que foi usada na respectiva operação, o restante do código aplica a mesma operação com a diferença de em vez de utilizar o método "sequencial" ele utiliza o método "aleatória", finalizando assim com a liberação de toda memória utilizada pelo método "liberarMatriz".

2.3 Linguagem de Programação e demais informações

A linguagem escolhida para a implementação do código foi Java, de maneira que se utiliza-se a linguagem principalmente abordada no segundo ano de formação para implementar o ciclo de instruções. Java é uma linguagem de alto nível desenvolvida pela Sun Microsystems, orientada a objetos e amplamente utilizada em vários ramos da programação. As bibliotecas da linguagem utilizadas para o funcionamento do código foram as bibliotecas "Arrays" e "Random" do pacote "java.util", que são usadas para manipular vetores e gerar números aleatórios, respectivamente.

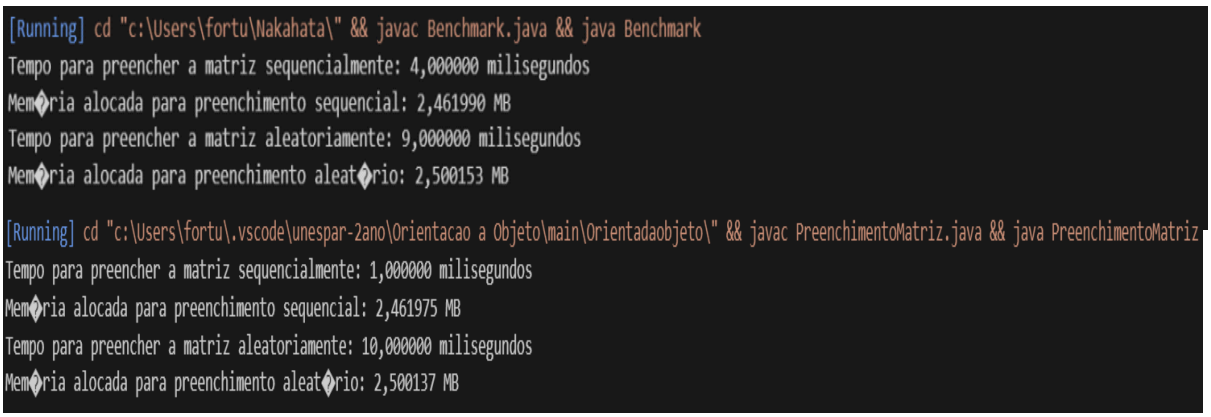
CAPÍTULO 3

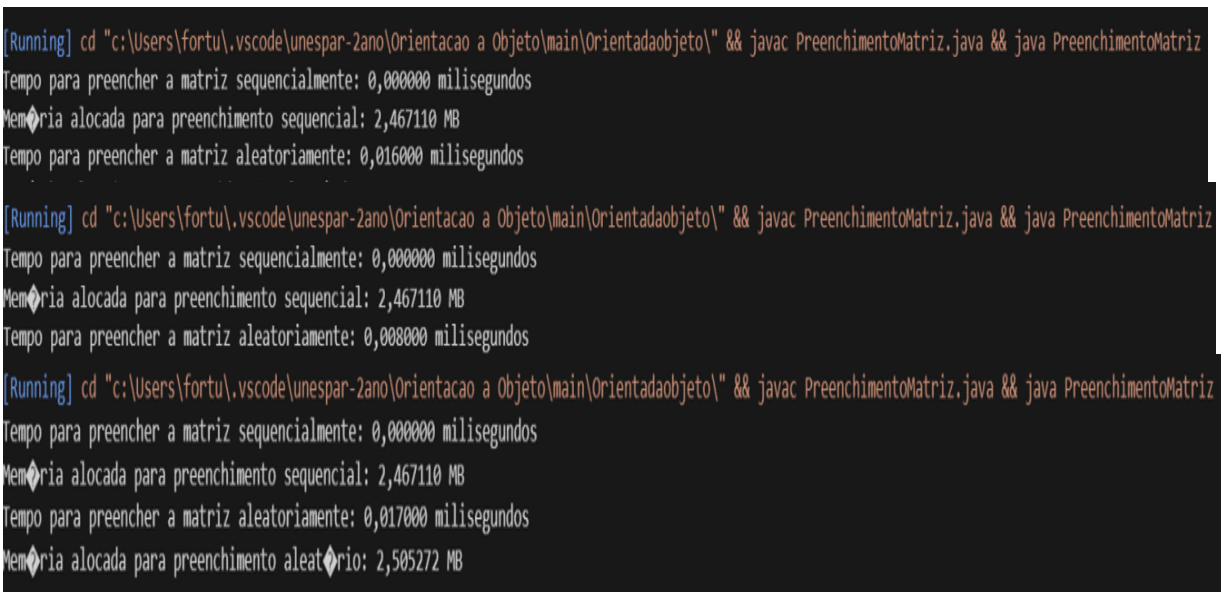
RESULTADOS

Mediante os objetivos apresentados, o resultado esperado seria o pleno funcionamento de um código que tem como função apresentar o desempenho computacional que nicha-se na operação de preenchimento matricial de modo sequencial e aleatório;

Pelos testes feitos tira-se a conclusão que no 1° teste o tempo de operação sequencial tende a ser maior como mostrados nas imagem 1 e 2, depois de 2 testes porém ele mantém uma certa constância apenas mostrando o tempo de operação no metódo aleatório como mostra nas figuras, concluindo que o tempo de operação tende a ser executado pelo menos 2 vezes menor que 0,003(tempo em milisegundos).

Já os testes de memória se mantém constantes em ambas operações tendo como maior valor registrado 2,47110 MB na maneira sequencial e 2,505272 MB na maneira aleatória sendo também os mais comuns como mostra nas imagens de teste.





Os seguintes testes foram feitos em computadores de 2 colegas distintos, pode-se analisar que os resultados de preenchimento matricial tiveram um tempo de resposta maior e memória alocada constante igual, podendo-se tirar a análise de uma capacidade computacional menor, tornando o objetivo do programa conclusivo.

```
[Done] exited with code=0 in 1.803 seconds

[Running] cd "c:\Users\palom\UnesparC\devc\2ºano\Arq Org Comp\" && javac BenchmarkG.java && java BenchmarkG
Tempo para preencher a matriz sequencialmente: 1,000000 milisegundos
Memoria alocada para preenchimento sequencial: 2,480019 MB
Tempo para preencher a matriz aleatoriamente: 35,000000 milisegundos
Memoria alocada para preenchimento aleatorio: 2,518181 MB

[Done] exited with code=0 in 1.949 seconds

[Running] cd "c:\Users\palom\UnesparC\devc\2ºano\Arq Org Comp\" && javac BenchmarkG.java && java BenchmarkG
Tempo para preencher a matriz sequencialmente: 4,000000 milisegundos
Memoria alocada para preenchimento sequencial: 2,480019 MB
Tempo para preencher a matriz aleatoriamente: 27,000000 milisegundos
Memoria alocada para preenchimento aleatorio: 2,518181 MB

[Done] exited with code=0 in 1.803 seconds

[Running] cd "c:\Users\palom\UnesparC\devc\2ºano\Arq Org Comp\" && javac BenchmarkG.java && java BenchmarkG
Tempo para preencher a matriz sequencialmente: 0,000000 milisegundos
Memoria alocada para preenchimento sequencial: 2,480019 MB
Tempo para preencher a matriz aleatoriamente: 17,000000 milisegundos
```

```
cd "/home/okagawa/Downloads/" && javac Benchmark.java && java Benchmark
❯ cd "/home/okagawa/Downloads/" && javac Benchmark.java && java Benchmark
Tempo para preencher a matriz sequencialmente: 4.000000 milisegundos
Memória alocada para preenchimento sequencial: 2.320541 MB
Tempo para preencher a matriz aleatoriamente: 75.000000 milisegundos
Memória alocada para preenchimento aleatório: 2.600594 MB
 ~/Downloads
```


CONCLUSÃO

Com base no que foi exposto anteriormente, o benchmark demonstrou que, ao comparar métodos de preenchimento de matriz, tanto o desempenho quanto os requisitos de memória são fatores significativos a serem considerados. Enquanto o método sequencial oferece uma abordagem simples e ordenada, ele demanda menos tempo de acordo com os testes feitos para completar a tarefa. Por outro lado, o método aleatório tende a ser mais lento, pois não requer uma ordem específica para o preenchimento da matriz, além de a natureza aleatória levar a variações no tempo de execução entre diferentes execuções do programa.

Além disso, é importante ressaltar que ambos os métodos apresentaram requisitos semelhantes de memória, indicando que a alocação de recursos não é significativamente afetada pela escolha do método de preenchimento.

Como mostrado nos testes este benchmark fornece uma base sólida para tomar decisões informadas sobre a escolha do método de preenchimento de matriz, levando

em conta tanto o desempenho quanto os requisitos de memória.

REFERÊNCIAS

[1] "Introduction to Algorithms" de Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest e Clifford Stein