COC472 - Computação de Alto Desempenho

Primeiro Trabalho

Universidade Federal
do Rio de Janeiro

- Pedro Tubenchlak Boechat - DRE 119065050
  - pedroboechat@poli.ufrj.br

Rio de Janeiro - RJ
8 de Agosto de 2021 (2021.1)

Introdução

O código-fonte desse trabalho está disponível no meu GitHub (`https://github.com/pedroboechat/UFRJ/tree/main/COC472/Trabalho%201`). Esse repositório contém:

- Duas pastas (*c* e *fortran*) contendo o código do programa principal nas linguagens C e Fortran;
- Dois scripts de compilação (*compile.bat* e *compile.sh*);
- Um programa Python que facilita a criação dos CSVs com o *benchmark* do programa principal com diferentes configurações (*runner.py*);
- Arquivos CSV contendo os dados de *benchmark*;
- Um Jupyter Notebook contendo a criação dos gráficos do *benchmark* (*analysis.ipynb*);
- Uma pasta contendo arquivos PNG dos gráficos gerados (*plots*);
- Este PDF e o PDF contendo os enunciados.

Segue abaixo a resolução do trabalho.

Questão 1

Enunciado: *Estime o tamanho máximo dos arrays A, x e b que podem ser alocados no seu sistema para realização da tarefa.*

Utilizando um computador com 128GiB de memória RAM e tendo aproximadamente 100GiB de memória disponível, foi calculado um valor de 110000 resolvendo a equação que iguala a quantidade de memória total disponível com as necessidades de alocação de memória do programa. Essa equação é dada pela fórmula $8 \cdot (n^2 + 2n) = \{\text{RAM DISPONÍVEL}\} \cdot 2^{30}$, onde n^2 é o espaço alocado para a matriz A, $2n$ é o espaço alocado para os vetores x e b, 8 é a quantidade de bits em um byte e 2^{30} é a quantidade de bytes em 1GiB.

Questão 2

Enunciado: *Os arrays a serem utilizados durante as operações devem ser inicializados com números aleatórios (A e x no caso acima).*

Conforme consta no código-fonte disponível no GitHub, as implementações em ambas as linguagens inicializam os arrays com valores aleatórios entre 0 e 100.

Questão 3

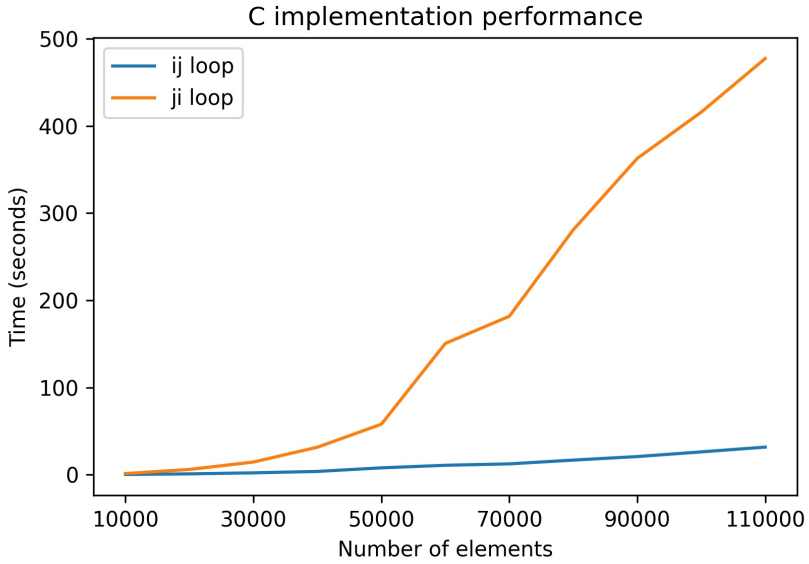
Enunciado: *Comece com um problema de tamanho pequeno e tente chegar ao tamanho máximo estimado no item 1. Contabilize o tempo para realização das operações para todos os tamanhos de sistema e para ambas as ordens de execução dos loops.*

Com a ajuda do *runner.py*, as implementações do programa pricipal em C e Fortan, iterando em linhas-colunas (*ij loop*) e colunas-linhas (*ji loop*), foram executadas uma vez para cada ARR_SIZE de 10000 a 110000, com *step* de 10000 e os dados de *benchmark* referentes ao tempo de execução foram exportados para arquivos CSV.

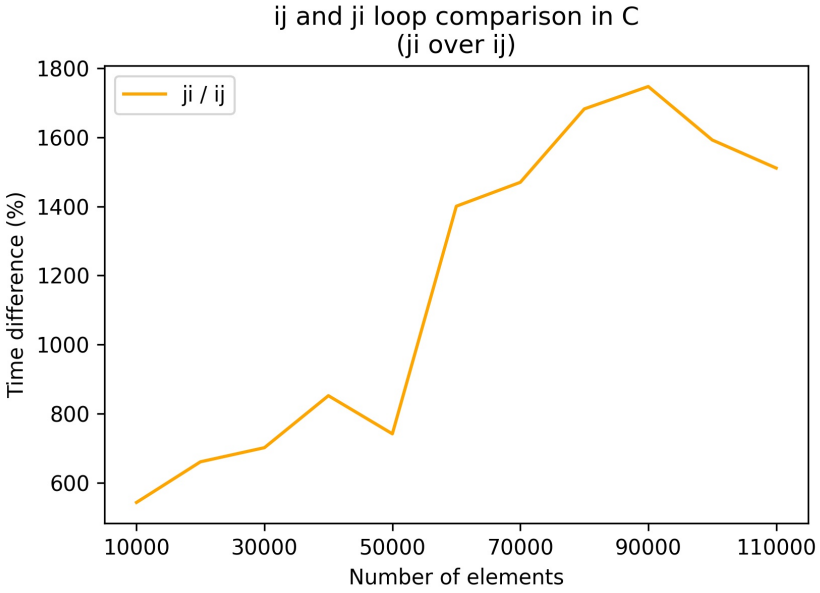
Questão 4

Enunciado: *Apresente curvas mostrando o tempo de execução para cada dimensão do problema e relacione estas curvas à complexidade computacional do produto matriz-vetor* ($\mathcal{O}(n^2)$).

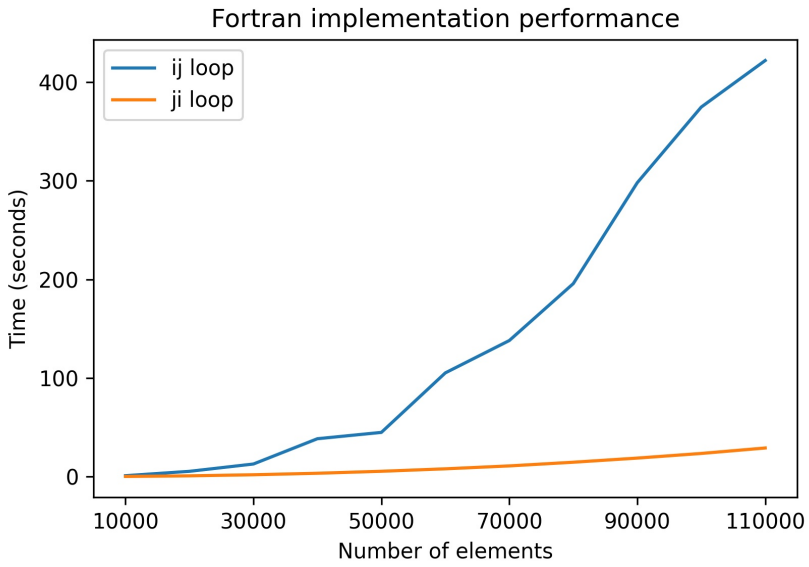
Com os dados obtidos a partir das instruções da *Questão 3*, foram gerados os seguintes gráficos:



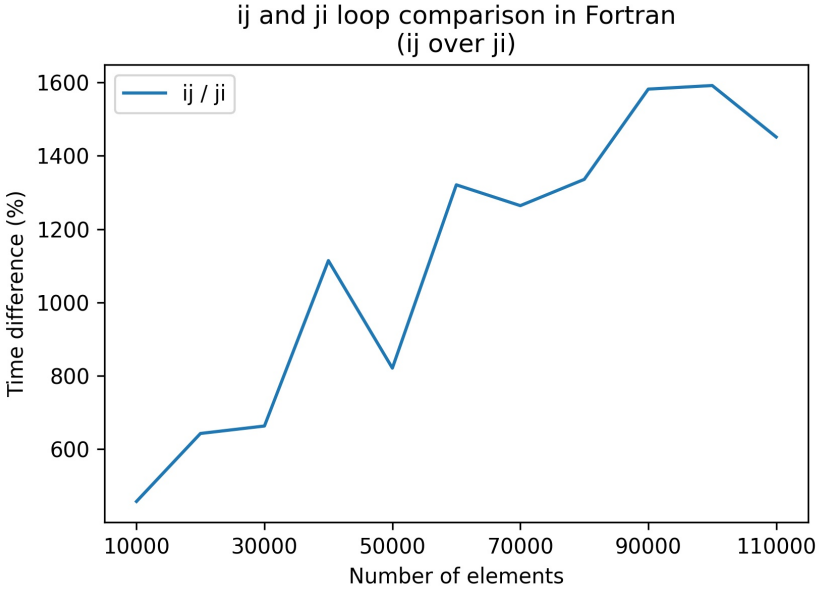
Benchmark da implementação na linguagem C



Comparação de quanto mais demorado a iteração colunas-linhas é em relação a iteração linhas-colunas na linguagem C



Benchmark da implementação na linguagem Fortran

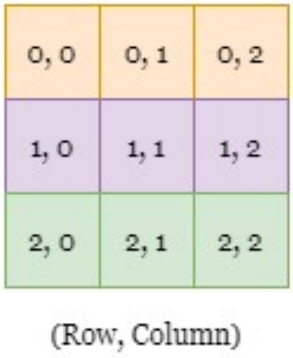


Comparação de quanto mais demorado a iteração linhas-colunas é em relação a iteração colunas-linhas na linguagem Fortran

Questão 5

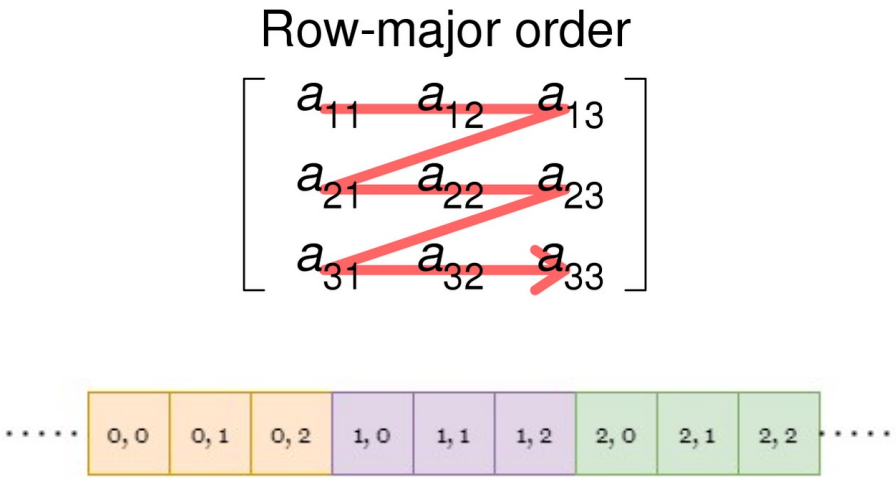
Enunciado: *Explique como o modo em que os arrays são armazenados nas duas linguagens afetam os resultados.*

As linguagens C e Fortran diferem no método de armazenamento de arrays multidimensionais em memória. Dado uma matriz 3x3 genérica:



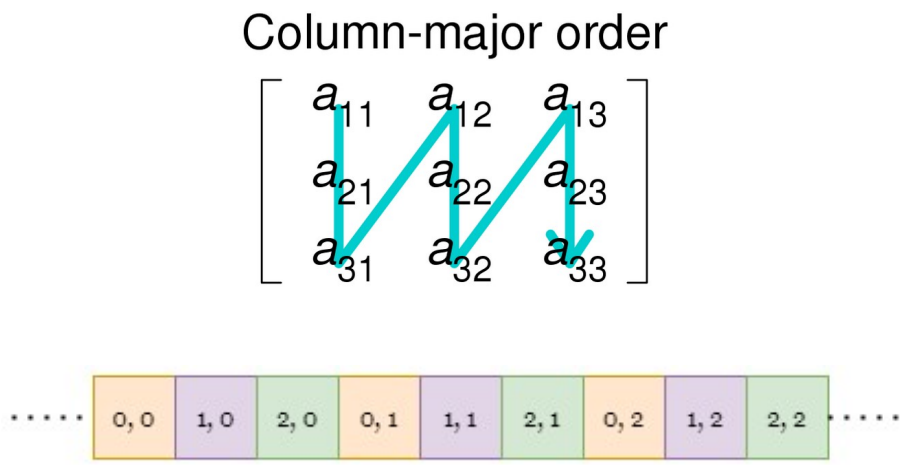
Matriz 3x3 genérica

A linguagem C utiliza a ordem principal de linha para armazenar essa matriz. Dessa forma, a alocação no endereço de memória segue a seguinte forma:



Ordem principal de linha

Já a linguagem Fortran utiliza a ordem principal de coluna para armazenar essa matriz. Dessa forma, a alocação no endereço de memória segue a seguinte forma:



Ordem principal de coluna

Em sistemas que usam cache do processador ou memória virtual, acessar elementos de um array é muito mais rápido se os elementos sucessivos estão armazenados em posições consecutivas em memória. Além disso, vários algoritmos que usam arrays multidimensionais procuram os elementos em ordens previsíveis. Assim, são justificadas as semelhanças e diferenças nos gráficos de *benchmark* de ambas as linguagens.

Referências

[1] Ordem principal de linha e de coluna - Wikipedia
`https://pt.wikipedia.org/wiki/Ordem_principal_de_linha_e_de_coluna`

[2] C vs Fortran memory order - Manik.cc
`https://manik.cc/2021/02/25/memory-order.html`

[3] Compact layouts (Array data structure) - Wikipedia
`https://en.wikipedia.org/wiki/Array_data_structure#Compact_layouts`