UFPEL

Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Curso de Bacharelado em Ciência da Computação
Curso de Bacharelado em Engenharia da Computação

Disciplina Conceitos de Linguagens de Programação

Comparação das linguagens Golang, Rust e C, com a implementação da Eliminação de Gauss

Felix Leonel Vasconcelos da Silva
"flvdsilva@inf.ufpel.edu.br"
Guilherme Dallmann Lima
"gdlima@inf.ufpel.edu.br"
Jelson Stoelben Rodrigues
"jsrodrigues@inf.ufpel.edu.br"
William Pedrolo Bourscheid
"wpbourscheid@inf.ufpel.edu.br"

Pelotas, maio de 2023

SUMÁRIO

1. INTRODUÇÃO

2. COMPARAÇÃO DAS LINGUAGENS

3. COMPARAÇÃO UTILIZANDO MÉTRICAS

4. COMPARAÇÃO DE DESEMPENHO

5. CONCLUSÕES

REFERÊNCIAS

1. Introdução

Este trabalho tem como objetivo compreender as características das linguagens de programação Rust e Golang, e realizar a comparação entre as linguagens Rust, Golang e C.

Foi realizada a implementação do algoritmo da eliminação gaussiana, inicialmente escrito em C nas linguagens Rust e Go.

2. COMPARAÇÃO DAS LINGUAGENS

As linguagens de programação Rust e Go são linguagens compiladas.

a. Comando de Iteração

O comando *for* do Rust é similar ao do Python: é dado um range de valores que a variável contadora deverá assumir nas iterações e alterações realizadas dentro do corpo do laço não alteram o conjunto de valores a serem percorridos, enquanto que o *for* do Go possui o comportamento similar ao do C: a variável pode ser alterada durante a iteração do comando e alterar o conjunto de valores a serem percorridos.

b. Representação de matriz

Na linguagem Go a matriz é similar ao C. Ao declarar matriz[N][M] será criado um bloco contíguo e as linhas serão mapeadas dentro deste bloco. A linguagem automaticamente calcula o endereço ao utilizar a notação *matriz[i][j]*. Podendo ser realizada a alocação de memória de forma contínua ou dinâmica.

Já na linguagem Rust, não existe nenhum tipo de abstração para o mapeamento de uma matriz sobre um bloco contíguo de memória e a forma mais simples de representar uma matriz é utilizando um vetor de vetores. Na implementação em Rust, foi utilizado a forma de vetor de vetores e também foi criado uma struct para realizar o mapeamento de uma matriz sobre um bloco contíguo de memória chamado de Array2D. Em Rust foi necessário sobrescrever o operador de indexação (matriz[][]) para calcular o endereço do dado dentro do bloco contíguo.

c. Controles de Fluxo

No Golang os comandos de controle de fluxo são similares ao do C, como por exemplo o *if*, *switch*, *continue* e *break*, entretanto há algumas diferenças, em Golang os comandos *if* e *switch*, aceitam uma instrução de inicialização, sendo comum para configurar a variável local.

Em Rust os comandos de controle de fluxo são *if* e *match.* O comando *if* padrão é igual ao funcionamento em C. O comando *match* é similar ao comando *switch* do C, com a exceção que todas as possibilidades devem ser tratadas, caso contrário o código não compila.

d. Chamadas de Função

Em Golang as funções não são similares ao C, pois em Golang *não é obrigatório* definir um nome para a função, apenas o tipo de dado retornado pela função ou método é obrigatório, o que difere da linguagem C, onde é preciso estar definindo um nome e o tipo de dado retornado pela função, também em Golang é possível a partir de uma função retornar mais de um valor, enquanto em C só é possível retornar apenas um valor, também em Golang ao chamar a função os valores da função são inicializados com zero, o que também difere da linguagem C, onde os valores da função não são inicializados com zero, o que pode resultar em lixo de memória para esses valores.

Em Rust os parâmetros de uma função podem ser passados por referência, referência mutável, ou por cópia. A passagem por cópia irá realizar uma cópia profunda dos dados quando o tipo tiver o *trait Clone* implementado, senão possuir, então será passado o ownership do dado para a função e se não for retornado no final da função, o dado será desalocado no término da função. Assim como em Go, é possível retornar mais de um valor na função ao utilizar tuplas.

e. Tipos de dados

Os tipos de dados em C, Golang e Rust são similares, entretanto em Golang possui o tipo de dado complex64 e complex128, que representa números complexos com precisão simples ou dupla, e também em Golang e Rust existe o tipo bool por padrão, enquanto em C não possui esse tipo de dado de forma nativa.

3. COMPARAÇÃO UTILIZANDO MÉTRICAS

A métrica utilizada para comparação entre as linguagens C, Golang e Rust, foi a dificuldade para se codificar e entendimento do código referente à implementação do método matemático da Eliminação de Gauss.

Como na linguagem Rust não existe abstrações para matrizes, foi necessário implementar uma struct para lidar com isto. As regras de ownership de Rust são difíceis de entender no início e é necessário muitas vezes repensar a forma do algoritmo para atingir o que é feito em outro código de outra linguagem. Porém Rust garante segurança de memória e não necessita de *garbage collector*.

Já o Golang é muito parecido com o C, o que diminui a dificuldade de programação, devido ao entendimento que já possuíamos sobre a linguagem C, onde a maior dificuldade encontrada para a conversão dos códigos, foi encontrar as bibliotecas equivalentes entre as linguagens, possibilitando obter acesso em Golang aos mesmos recursos utilizados na codificação do programa em C.

Comparando o número total de linhas utilizadas, em Rust foram necessárias 154 linhas de código, enquanto que em Go apenas 73 linhas foram necessárias. Isto se deu porque, em Rust foi necessário implementar a abstração para utilizar uma matriz mapeada em um bloco de memória contínuo, enquanto que em Go não foi necessário. E também porque existem duas versões de representação de matriz implementadas em Rust, utilizando o mapeamento de uma matriz em cima de um bloco contíguo e utilizado um vetor de vetores.

4. COMPARAÇÃO DE DESEMPENHO

Os testes foram realizado em um computador com as seguintes características:

Processador *R5 5600x*, com *32GB de memória RAM* e sistema operacional *Linux Mint 21.1*.

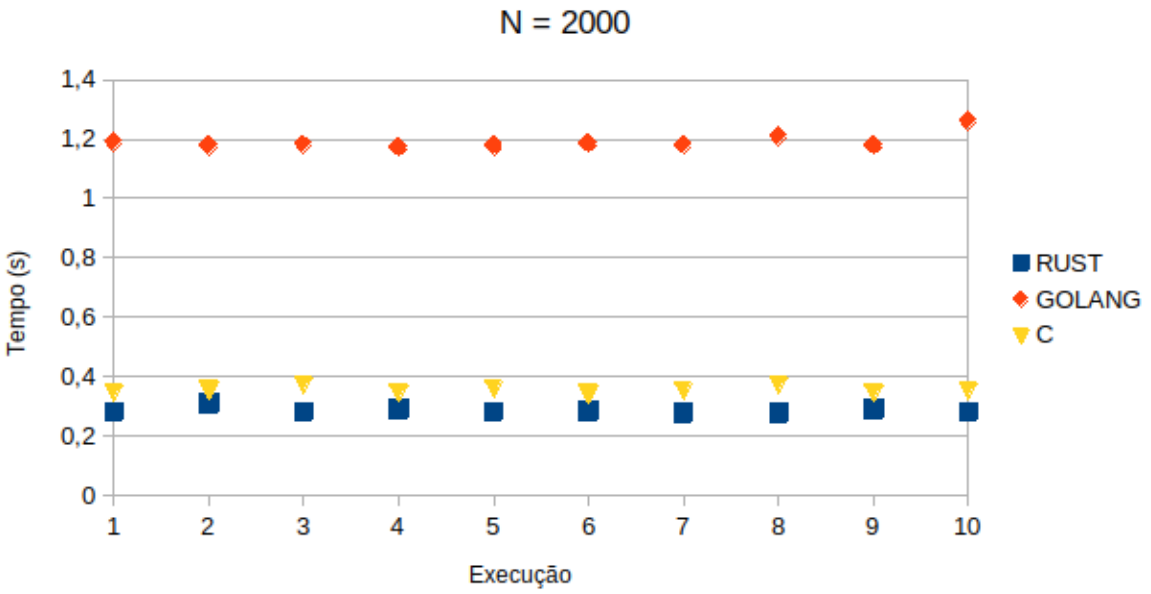
O tempo foi medido apenas na execução da solução do problema, sem contar a criação dos valores e exibição dos resultados.

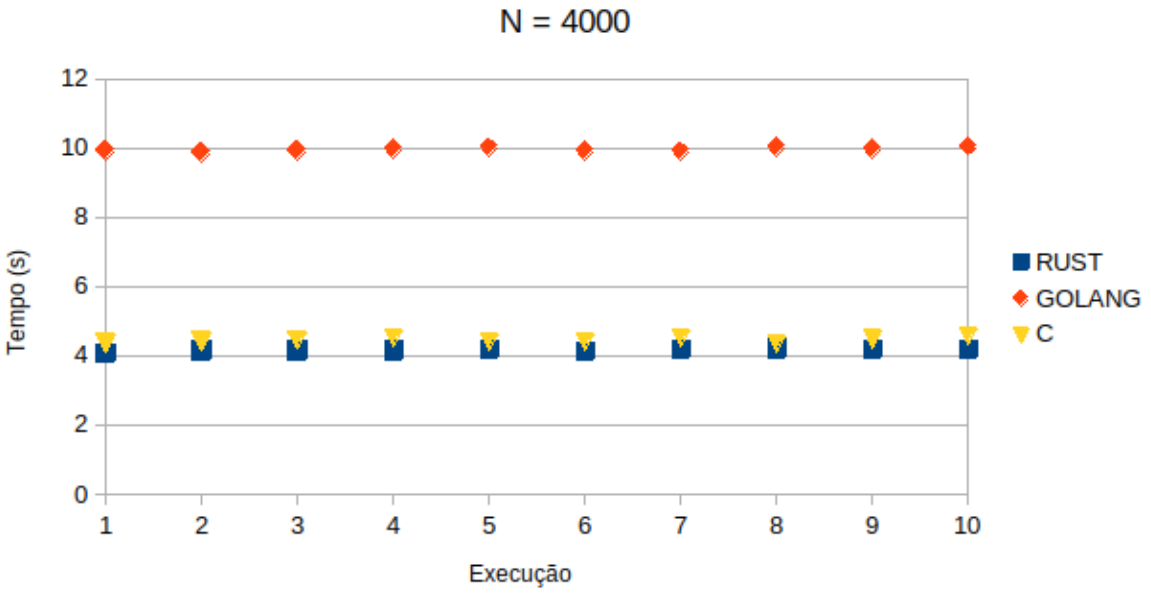
Todos os códigos foram compilados com flags de otimização e os tamanhos testados foram para matriz de ordem 2000 e 4000. Foram realizadas 10 execuções de cada linguagem em cada tamanho de matriz, sendo que foi executado de forma intercalada, não foi executado as 10 vezes em Rust para então 10 execuções em Go e 10 execuções em C, mas a primeira amostra foi coletada executando em Rust, depois em Go depois em C, para então a segunda amostra ser coletada utilizando a mesma ordem. Na linguagem Rust foi utilizado a versão de representação de matriz de mapeamento sobre uma região de memória contínua, que foi implementada do zero, por ela ter o mesmo comportamento das outras linguagens.

Todas as implementações são sequenciais.

Tabela 1. Tempo de execução em segundos para solucionar um sistema linear utilizando o método de Gauss, comparando as linguagens Rust, Golang e C.

Execução	**Rust**		**Golang**		**C**	
Ordem	**N = 2000**	**N = 4000**	**N = 2000**	**N = 4000**	**N = 2000**	**N = 4000**
1	0,28041804	4,08844400	1,18963198	9,91828114	0,34463300	4,38985000
2	0,31071037	4,16220240	1,17718208	9,88598786	0,35619300	4,43183000
3	0,28213516	4,14745040	1,18194146	9,91820600	0,37342200	4,46482000
4	0,29109250	4,16060800	1,17435372	9,98233354	0,34601700	4,53002000
5	0,28046720	4,17656000	1,17830686	10,01663547	0,36012400	4,39371000
6	0,28304997	4,14564200	1,18713943	9,93694302	0,34363400	4,41168000
7	0,27882624	4,19848870	1,17916897	9,91252340	0,35383800	4,50584000
8	0,27849160	4,20411350	1,20999431	10,03310489	0,37310500	4,35057000
9	0,29227513	4,19109540	1,17954570	9,97959969	0,34566700	4,48387000
10	0,28139308	4,18924570	1,26209287	10,05621805	0,35394300	4,57516000
Média	**0,28588593**	**4,16638501**	**1,19193574**	**9,96398331**	**0,35505760**	**4,45373500**





5. CONCLUSÕES

Analisando os resultados percebemos que houve pouca diferença entre as implementações em C e Rust e que Go foi mais lenta.

Isto se dá pois tanto C como Rust são linguagens voltadas a performance. Go teve um desempenho inferior, pois utiliza garbage collector em tempo de execução e o runtime da linguagem é maior se comparado com Rust e C.

As diferenças entre as linguagens C e Rust pode ser explicada também pela eficiência das otimizações do compilador de cada linguagem na geração de código executável.

REFERÊNCIAS

Rust Programming Language. (2021). Rust Documentation (Beta). Acesso em: 01 de maio de 2023, de https://doc.rust-lang.org/beta/.

Rob Pike. Effective Go. Disponível em: https://go.dev/doc/effective_go. Acesso em: 01 de maio de 2023.

devdocs.io. (s.d.). C documentation Disponível em: https://devdocs.io/c/. Acesso em: 01 de maio de 2023.

Mendonça, G. (n.d.). Gaussian elimination with Pthreads and OpenMP. Acesso em: https://github.com/gmendonca/gaussian-elimination-pthreads-openmp/blob/master/gauss.c